ISO 29990:2010

NATIONAL INFOCOMM COMPETENCY FRAMEWORK

NUS National University of Singapore | ISS INSTITUTE OF SYSTEMS SCIENCE

Text Processing using Machine Learning

Machine Translation

Liling Tan

13 Mar 2019

Overview

Lecture

- Sentence Representation

Hands-on

- PyTorch LMs

Phrase-Based Machine Translation

PBMT: The Mathematics of MT

The objective of the MT system is to find the best translation $\hat{\boldsymbol{t}}$ that maximizes the translation probability $\mathrm{p}(\boldsymbol{t}|\boldsymbol{s})$ given a source sentence $\boldsymbol{s}$; mathematically:

$$\hat{t} = \underset{t}{argmax}\ p(t|s) \tag{3}$$

Applying the Bayes' rule, we can factorized the $\mathrm{p}(\boldsymbol{t}|\boldsymbol{s})$ into three parts:

$$p(t|s) = \frac{p(t)}{p(s)} p(s|t) \tag{4}$$

PBMT: The Mathematics of MT

Substituting our $\mathrm{p}(\boldsymbol{t}|\boldsymbol{s})$ back into our search for the best translation $\hat{\boldsymbol{t}}$ using *argmax*:

$$\begin{aligned}\hat{t} &= \underset{t}{argmax}\ p(t|s) \\ &= \underset{t}{argmax}\ \frac{p(t)}{p(s)}p(s|t) \\ &= \underset{t}{argmax}\ p(t)p(s|t)\end{aligned} \tag{5}$$

We note that the denominator $\mathrm{p}(\boldsymbol{s})$ can be dropped because for all translations the probability of the source sentence remains the same and the *argmax* objective optimizes the probability relative to the set of possible translations given a single source sentence.

PBMT: The Mathematics of MT

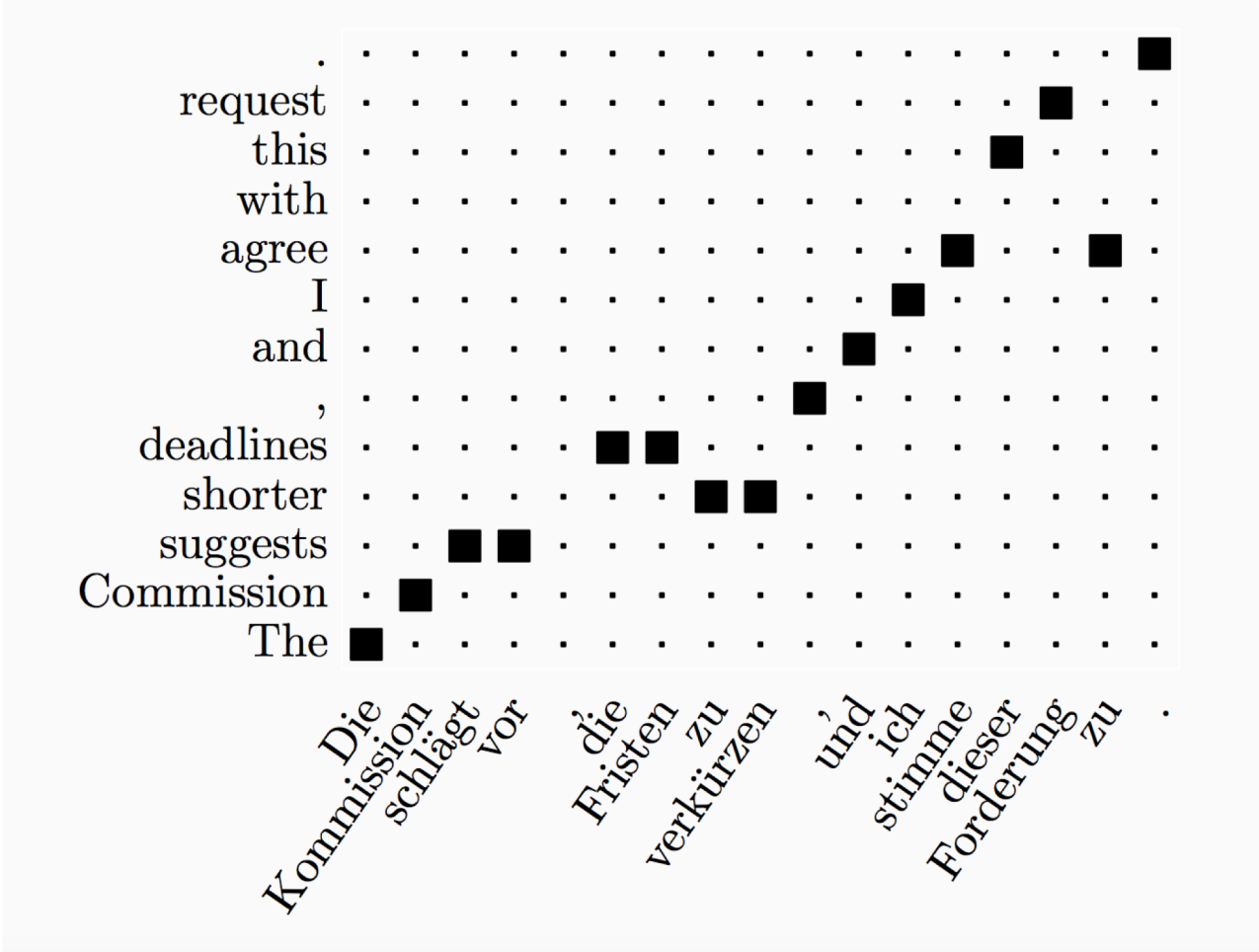
Extending the noisy channel model, Och & Ney (2002) simplified the integration of additional model components using the *log-linear model*. The model defines feature functions $h(x)$ with weights λ in the following form:

$$P(x) = \frac{exp(\sum_{i=1}^{n} \lambda_i h_i(x))}{Z} \tag{6}$$

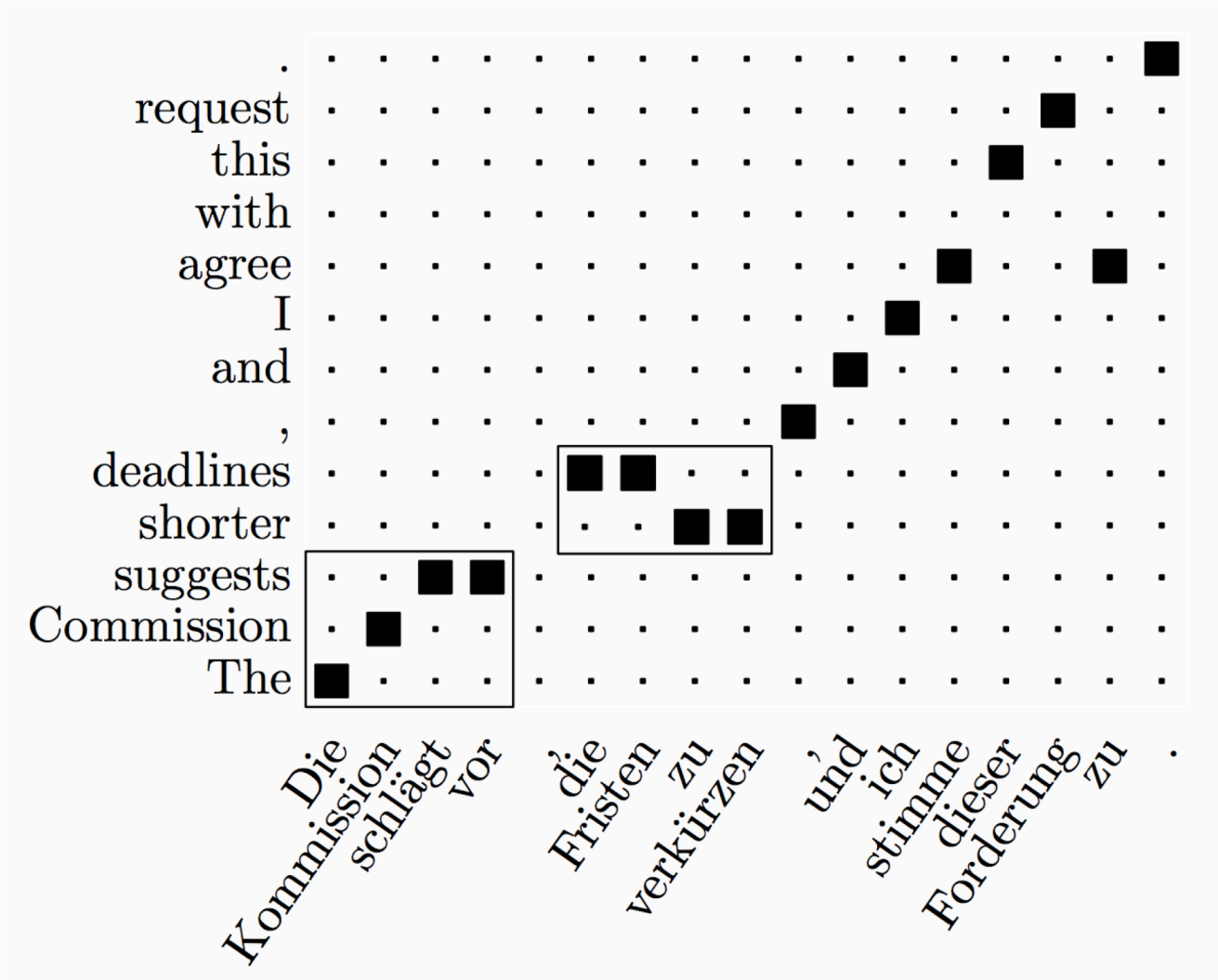
where the normalization constant Z turns the numerator into a probability distribution. In the case of a simple model that contains the two primary features from the noisy channel model, we define the components as such:

$$\begin{aligned} h_1(x) &= p(t) \\ h_2(x) &= p(s|t) \end{aligned} \tag{7}$$

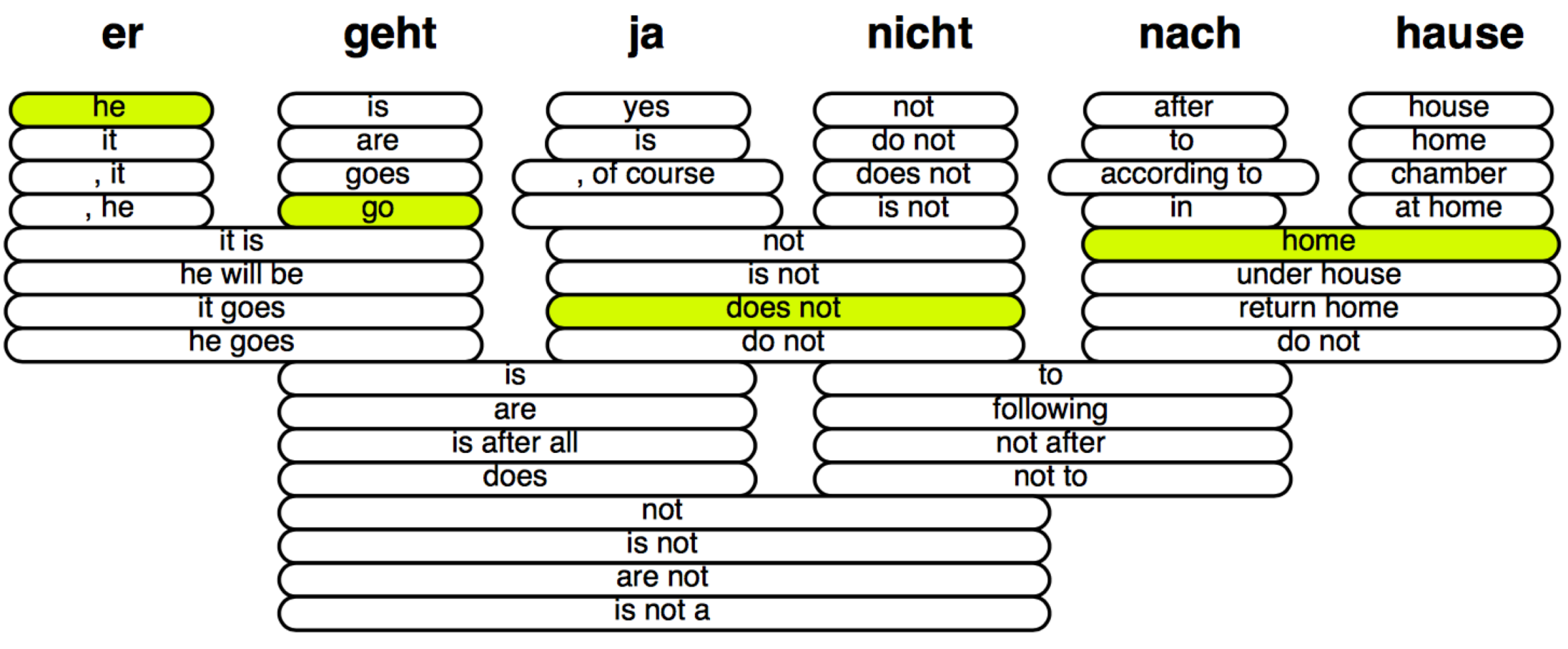
PBMT: Word Alignment (IBM Models)



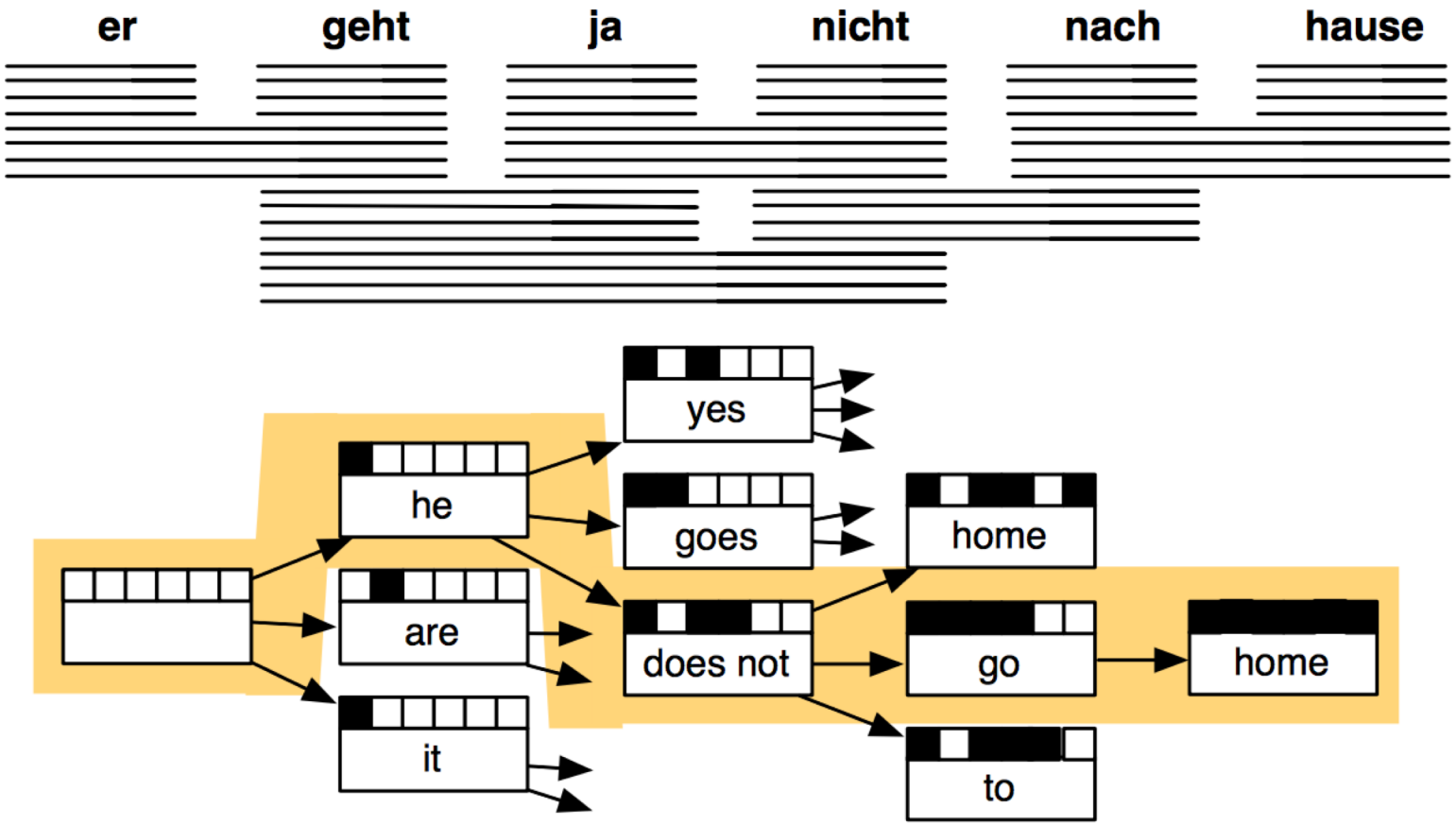
PBMT: Word to Phrase



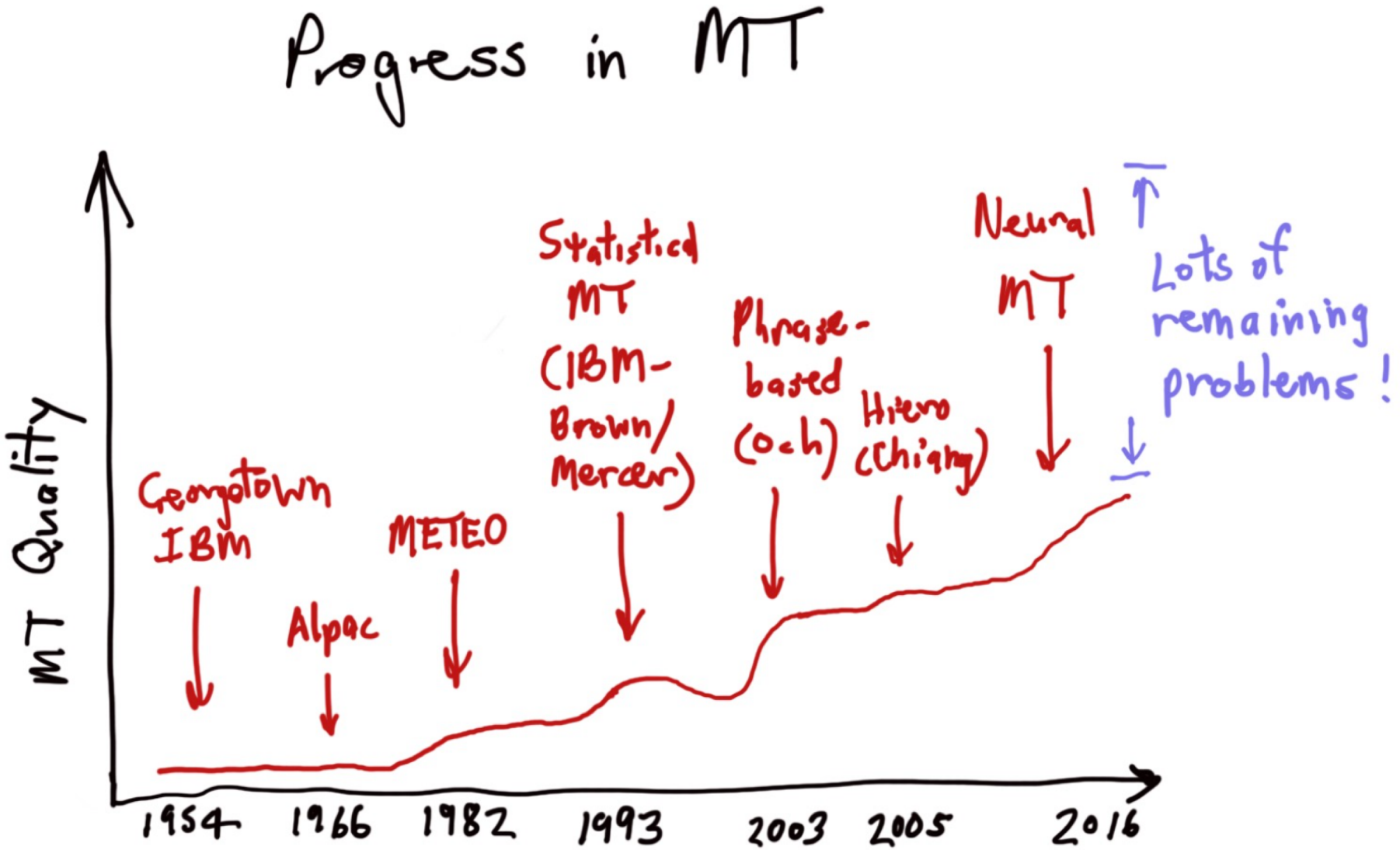
PBMT: Phrase Table



PBMT: Language Model and Decoding



PBMT -> Neural MT



PBMT -> Neural MT

"**Statistical MT** systems, built by **hundreds** of engineers over many **years**, outperformed by **NMT** systems trained by a **handful** of engineers in a **few months**" – See (2019)

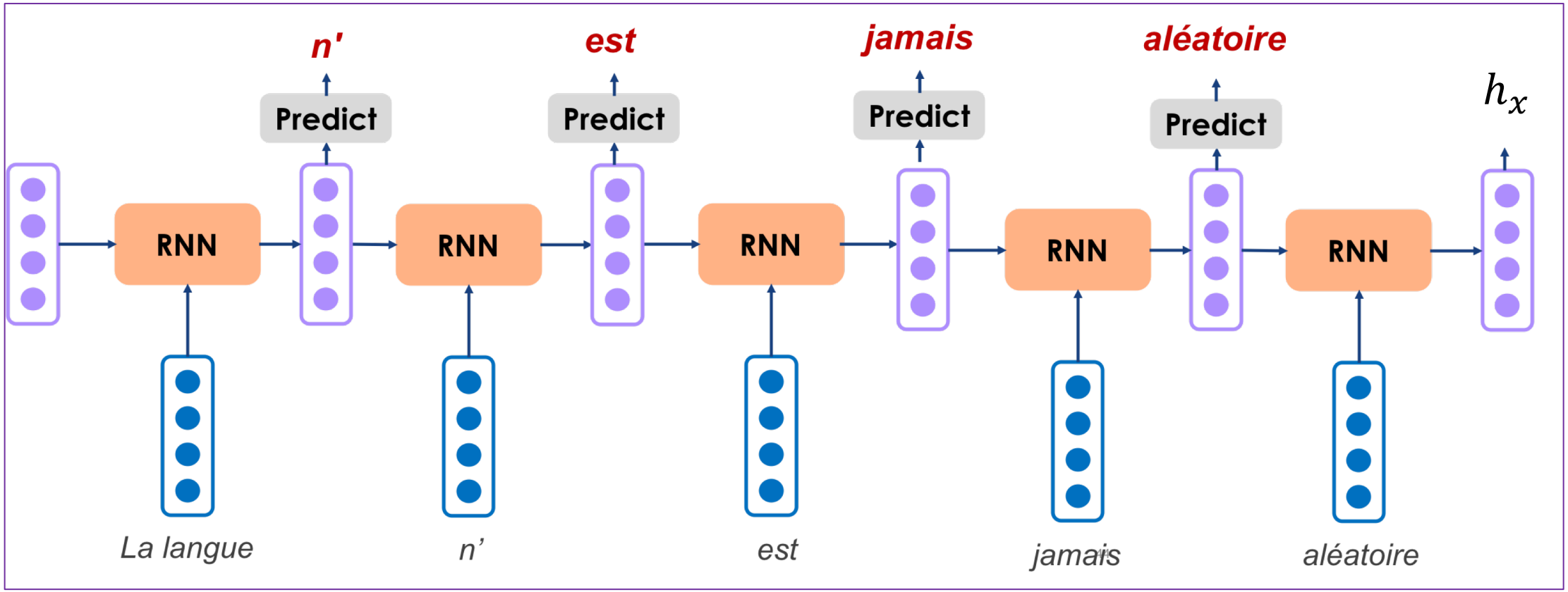
PBMT vs Neural MT

PBMT	NMT
− Local context	+ Global context
○ Independent models	+ Global optimization
− LM one of many models	+ Generation guided by LM
+ Coverage constraints	− Over-/under-generation
+ Model introspection	− "Black box" approach
− Model size	+ Model size
	− Misspellings/new words

Neural Machine Translation

Language Model

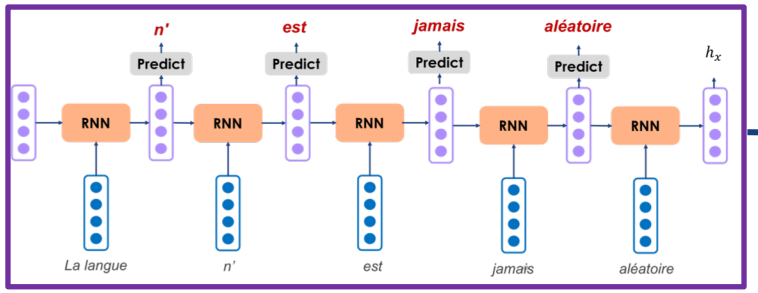
- Language Model encode an input sentence into a h_x



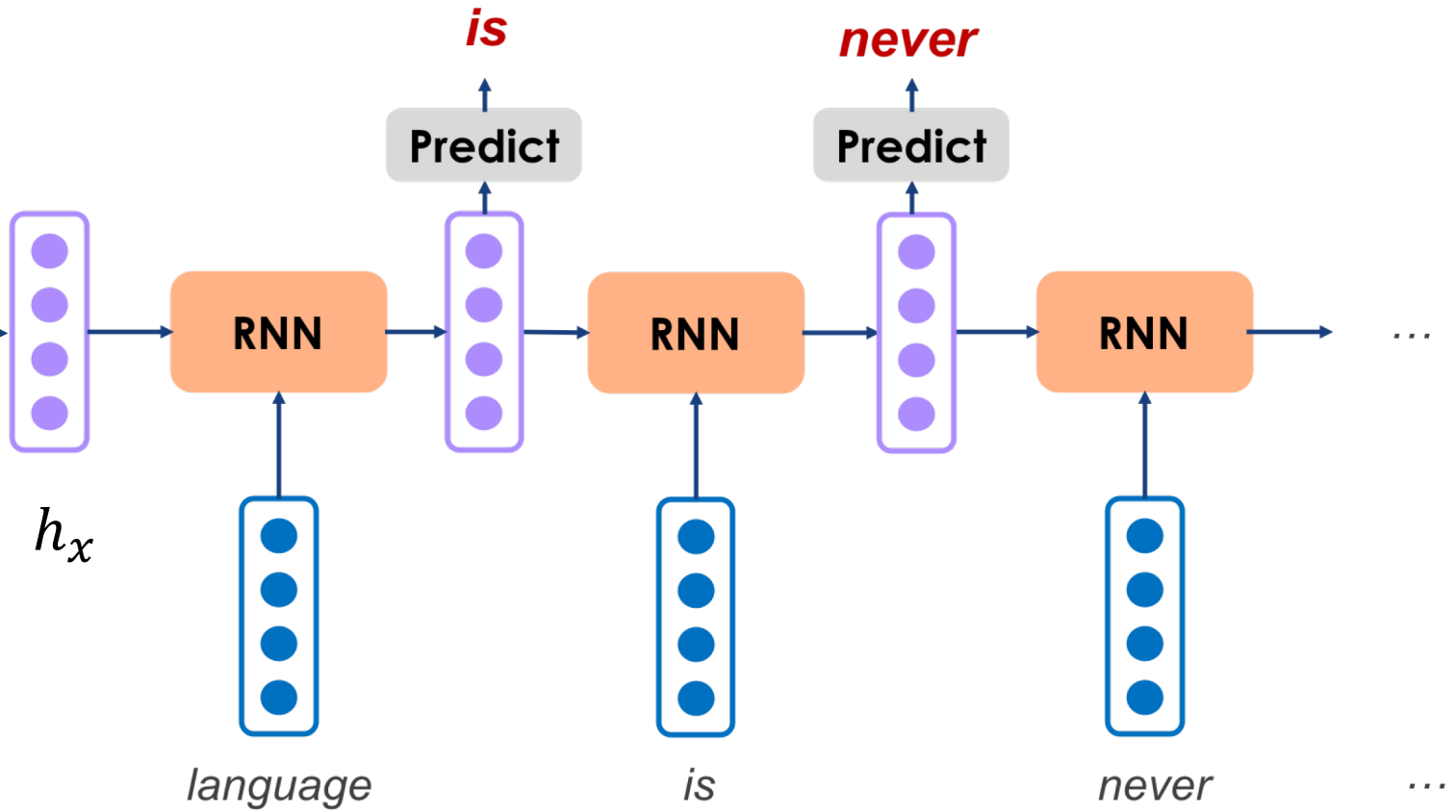
Encoder Model

Sequence to Sequence Learning with NN

- Take final hidden state of an encoder model, feed it as the start state of a decoder model (Sutskever et al. 2014)

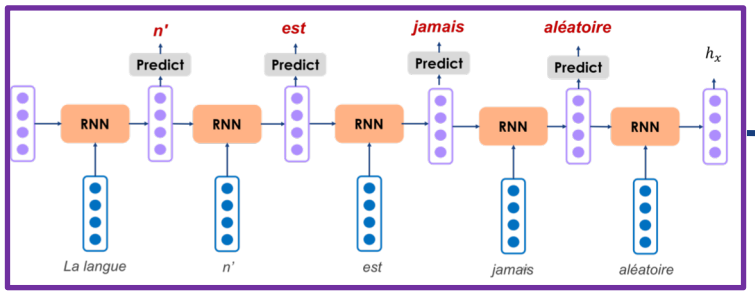


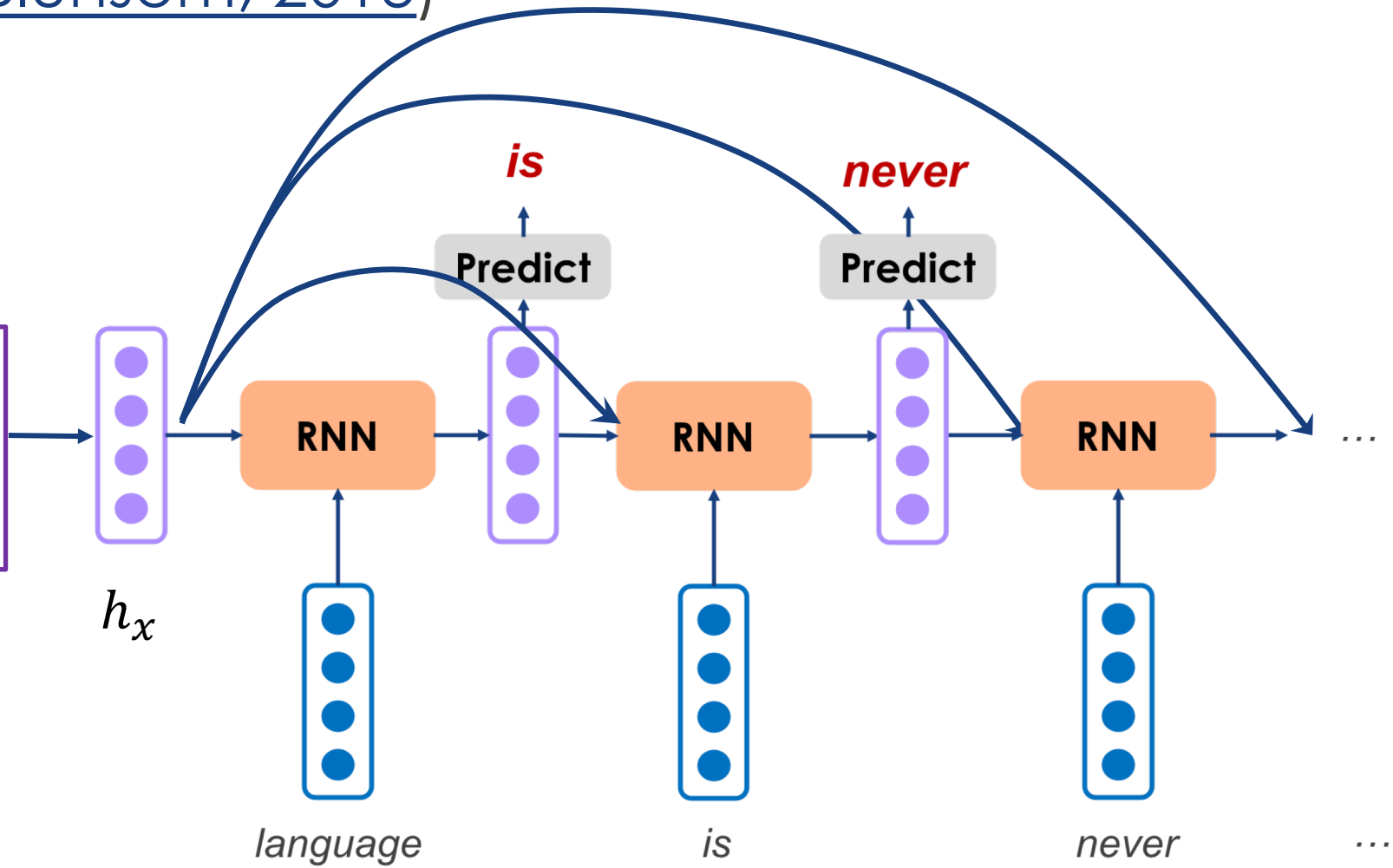
Encoder Model



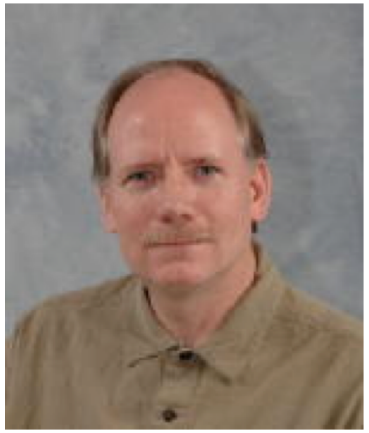
Recurrent Continuous Translation Models

- Add the encoded hidden state at every decoder time step (Kalchbrenner & Blunsom, 2013)





Sentence Representation



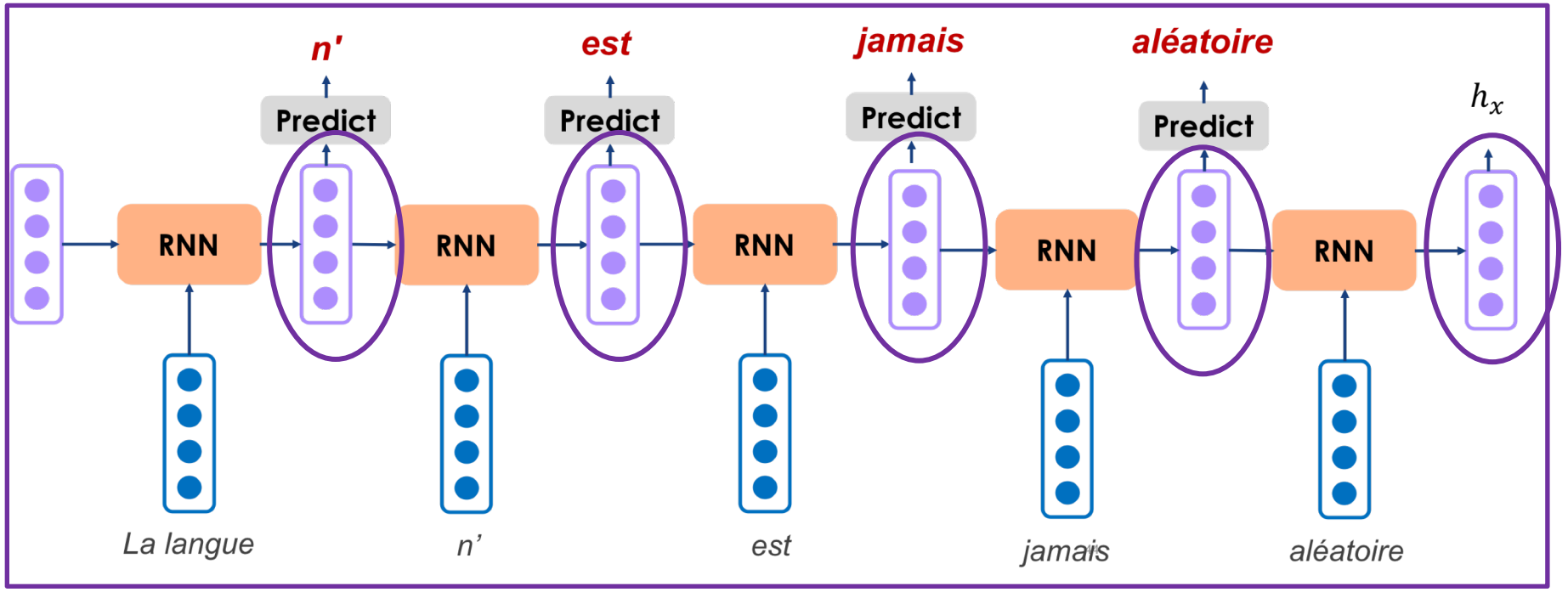
"You can't cram the meaning of a whole !@#$-ing sentence into a single %^&-ing vector!"*

– Raymond Mooney

Can't queeze sentence *into a %^&*-ing vector*

"You can't cram the meaning of a whole !@#$-ing sentence into a single %^&-ing vector!"*

– Raymond Mooney



What if instead of taking just h_x, we take **all the hidden states**?

Attention: Jointly Learning to Align and Translate

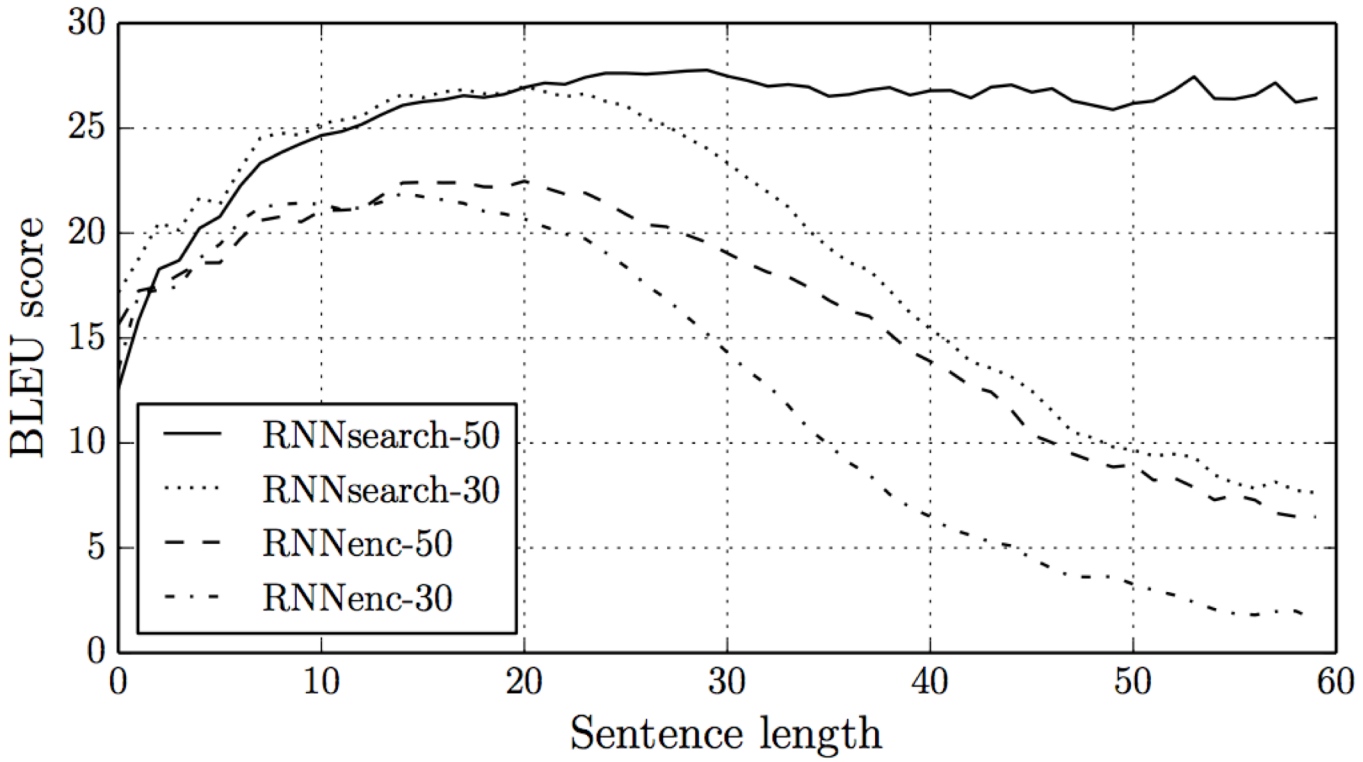


Figure 2: The BLEU scores of the generated translations on the test set with respect to the lengths of the sentences. The results are on the full test set which includes sentences having unknown words to the models.

Bahdanau et al. (2015)

Machine Translation Evaluation

Bilingual Evaluation Understudy (Papineni et al. 2002)

Papineni et al. (2002) originally define BLEU n-gram precision p_n by summing the n-gram matches for every hypothesis sentence S in the test corpus C:

$$p_n = \frac{\sum_{S \in C} \sum_{ngram \in S} Count_{matched}(ngram)}{\sum_{S \in C} \sum_{ngram \in S} Count(ngram)} \quad (1)$$

BLEU is a precision based metric; to emulate recall, the brevity penalty (BP) is introduced to compensate for the possibility of high precision translation that are too short. The BP is calculated as:

$$BP = \begin{cases} 1 & \text{if } c > r \\ e^{1-r/c} & \text{if } c \leq r \end{cases} \quad (2)$$

where c and r respectively refers to the length of the hypothesis translations and the reference translations. The resulting system BLEU score is calculated as follows:

$$\text{BLEU} = \text{BP} \times \exp\left(\sum_{n=1}^{N} w_n \log p_n\right) \quad (3)$$

where n refers to the orders of n-gram considered for p_n and w_n refers to the weights assigned for the n-gram precisions; in practice, the weights are uniformly distributed.

Fear the BLEU side…

```
alvas@ubi:~/git/mosesdecoder/scripts/generic$ cat hyp.txt
foo bar
alvas@ubi:~/git/mosesdecoder/scripts/generic$ cat ref.txt
foo bar
alvas@ubi:~/git/mosesdecoder/scripts/generic$ perl multi-bleu.perl ref.txt < hyp.txt
BLEU = 0.00, 100.0/100.0/0.0/0.0 (BP=1.000, ratio=1.000, hyp_len=2, ref_len=2)
```

Awkward Disparity of BLEU

"Conventional" wisdom:

– **Lower BLEU** not necessarily worse translation

– **Higher BLEU** = better translation

But is **higher BLEU** = better translation true?

Awkward Disparity of BLEU

Source:
T용융(DSC) = 89.9℃; T결정화(DSC) = 72℃(5℃/분에서DSC로측정).

Hypothesis:
Ｔｍｅｌｔ（ＤＳＣ）＝７２℃（５℃／分でＤＳＣ測定（ＤＳＣ）＝89．９結晶化度（Ｔ）。

Baseline:
Ｔ溶融（ＤＳＣ）＝８９．９℃；Ｔ結晶化（ＤＳＣ）＝７２℃（５℃／分でＤＳＣで測定）。

Reference:
Ｔｍｅｌｔ（ＤＳＣ）＝８９．９℃；Ｔｃｒｙｓｔ（ＤＳＣ）＝７２℃（５℃／分でＤＳＣを用いて測定）。

Source/Reference English Gloss:
Tmelt (DSC) = 8 9. 9 °C; Tcryst (DSC) = 7 °C (measured using DSC at 5 °C / min)

Hypothesis	Baseline
P_1 : 90.0	P_1 : 84.2
P_2 : 78.9	P_2 : 66.7
P_3 : 66.7	P_3 : 47.1
P_4 : 52.9	P_4 : 25.0
BP: 0.905	BP: 0.854
BLEU: 64.03	BLEU: 43.29
HUMAN: -5	HUMAN: 0

Machine Translation Human Evaluation

Human Evaluation: Adequacy

- 1 = Complete nonsense
- 2 = Very little meaning of the source sentence is captured
- 3 = Some meaning of the source sentence is captured
- 4 = Almost all meaning is captured
- 5 = Perfect translation of meaning

Human Evaluation: Fluency

- 1 = Incomprehensible
- 2 = Disfluent
- 3 = Acceptable
- 4 = Good
- 5 = Flawless

Human Evaluation: Post-Editability

1 = Easier to translate from scratch than to edit

2 = Requires the same time to edit as translating from scratch

3 = Requires some editing, easier to edit than translate from scratch

4 = Requires light editing

5 = Requires no editing

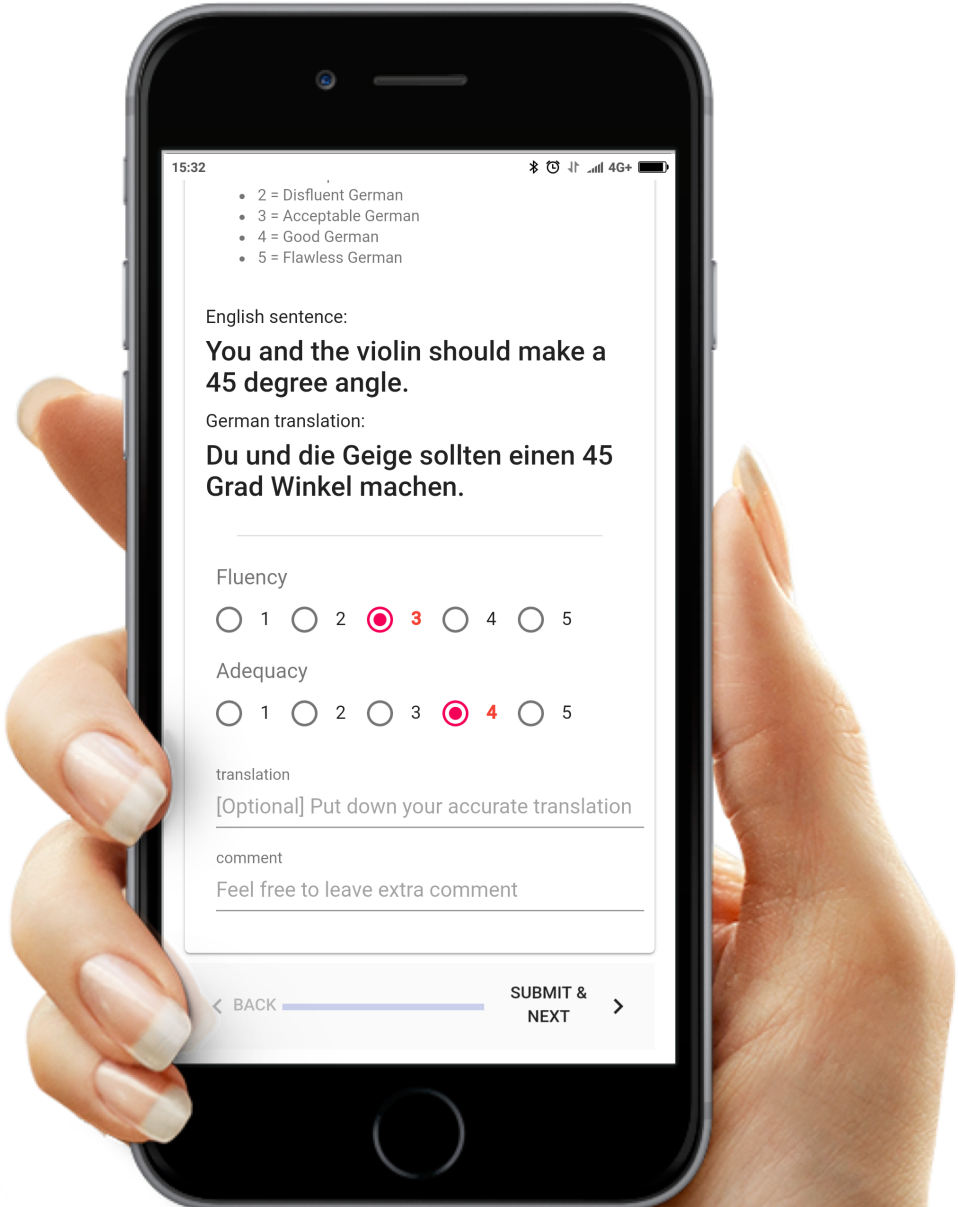
Rakuten Nimbus: MT Evaluation Platform

Good

Clear absolute score

Relatively fast evaluation

*Comparing against system doesn't require too many overlaps



Bad

Difficult to distinguish between fluency and adequacy

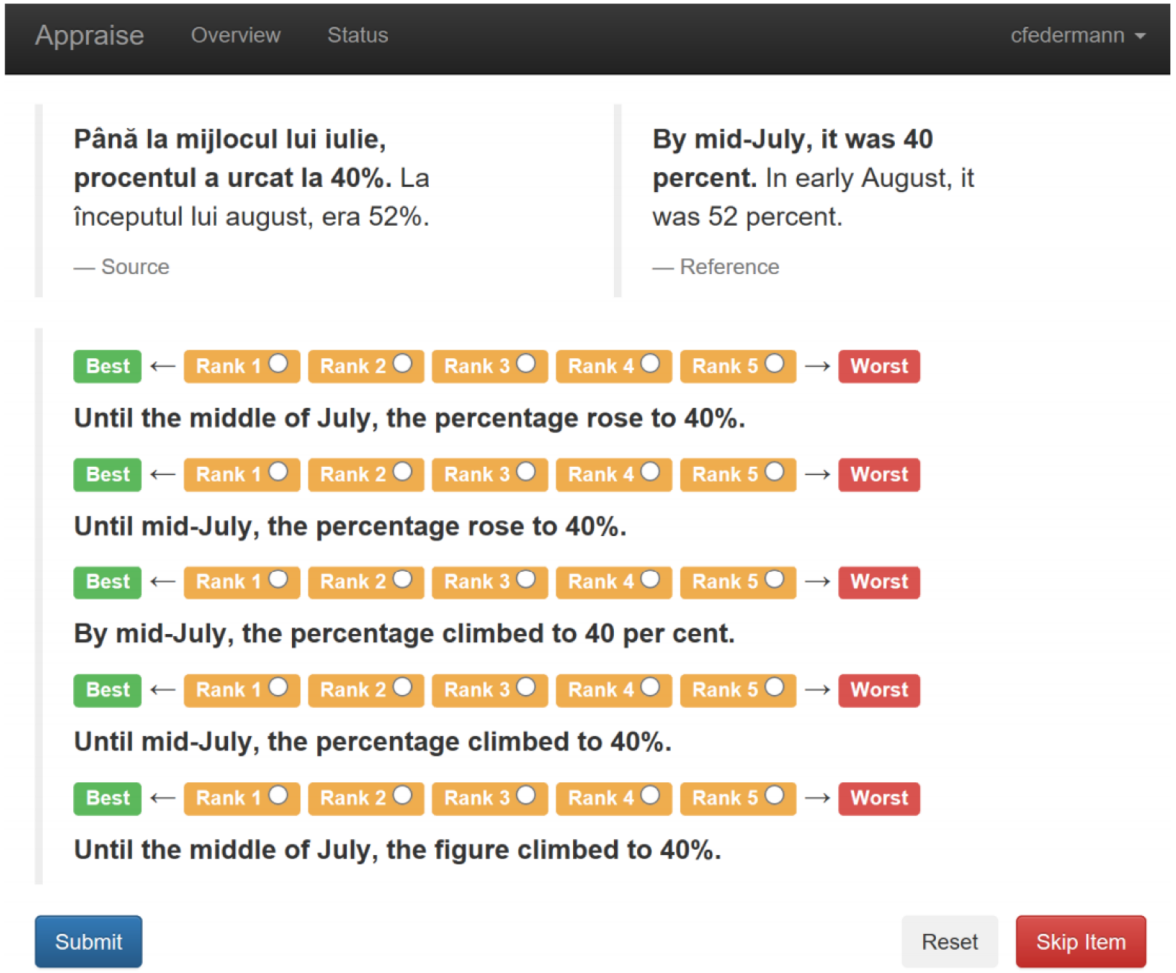
Nonsense still scoring as a 1 minimally

Humans can be inconsistent

Human Evaluation: Relative Ranking

Good

- HIT size: 3 x 5
- Relatively fast
- Skip-able
- Mental context



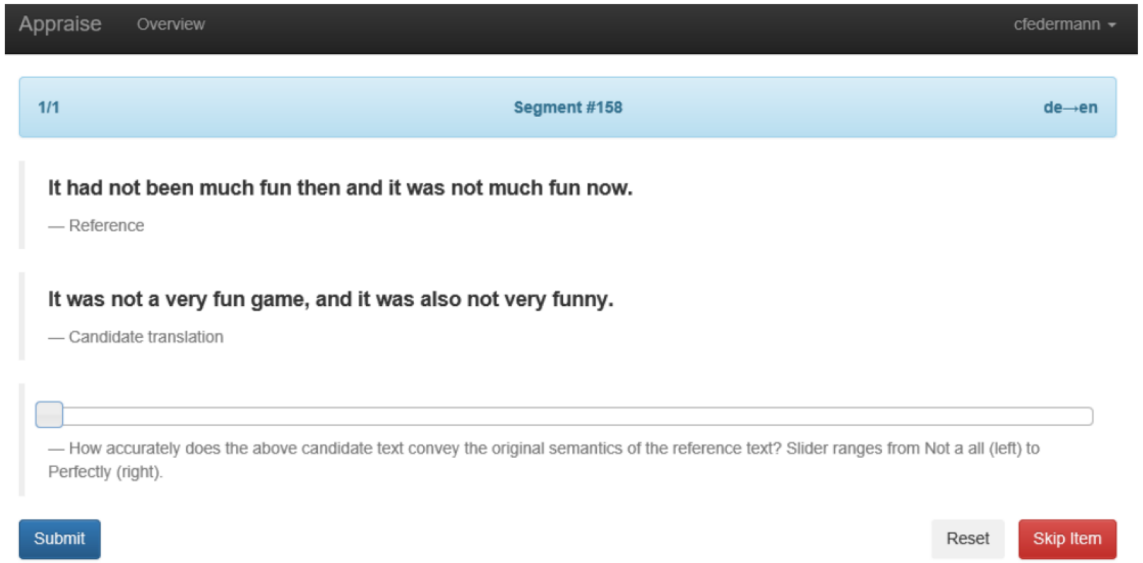
Bad

- Quadratic cost
- Cognitive load
- Long sentences
- Only relative deltas
- No absolute scores

Human Evaluation: Direct Assessment

Good

- Linear cost
- Cognitive load
- Absolute scores
- Long sentences



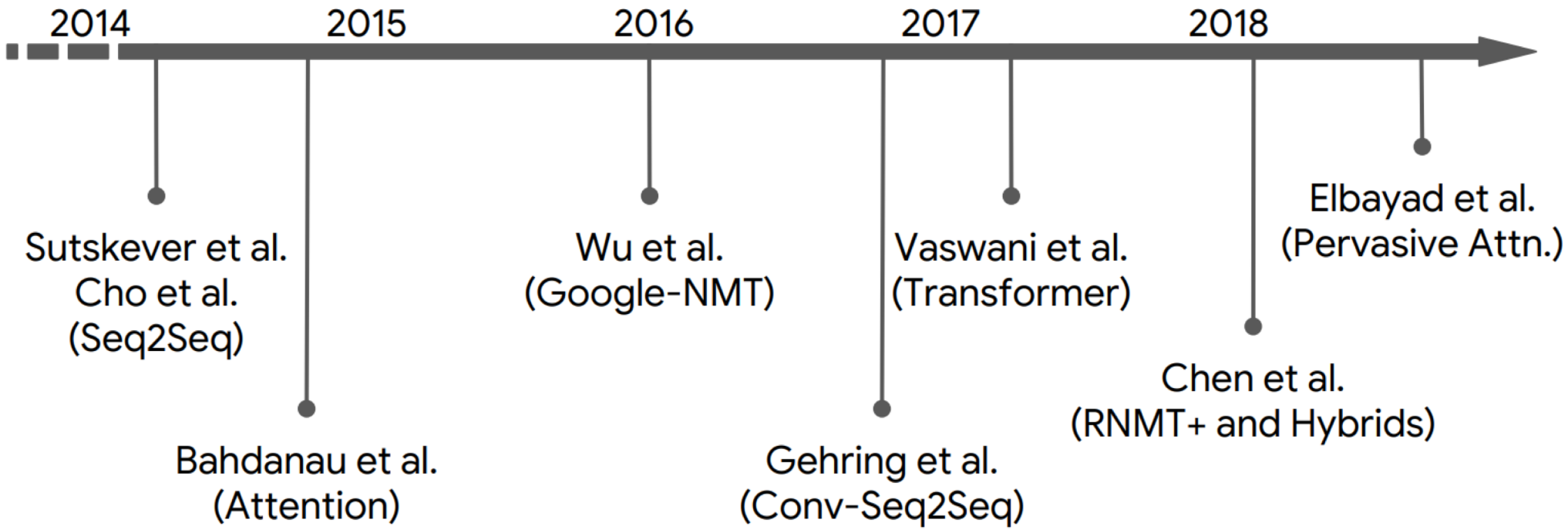
Bad

- HIT size: 100 x 1
- Comparatively slow
- Fuzzy mental context
- High loss for crowd

State-of-Art Neural NMT (in 2018)

Disclaimer: Most of the following slides in this section comes from Firat (2018) talk at MT Marathon

History of Neural MT (NMT)



Slide from Firat (2018) of Google AI

GNMT - Wu et al.

- Core Components:
  - RNNs
  - Attention (Additive)
  - biLSTM + uniLSTM
  - Deep residuals
  - Async Training
- Pros:
  - De facto standard
  - Modelling state space
- Cons:
  - Temporal dependence
  - Not enough gradients

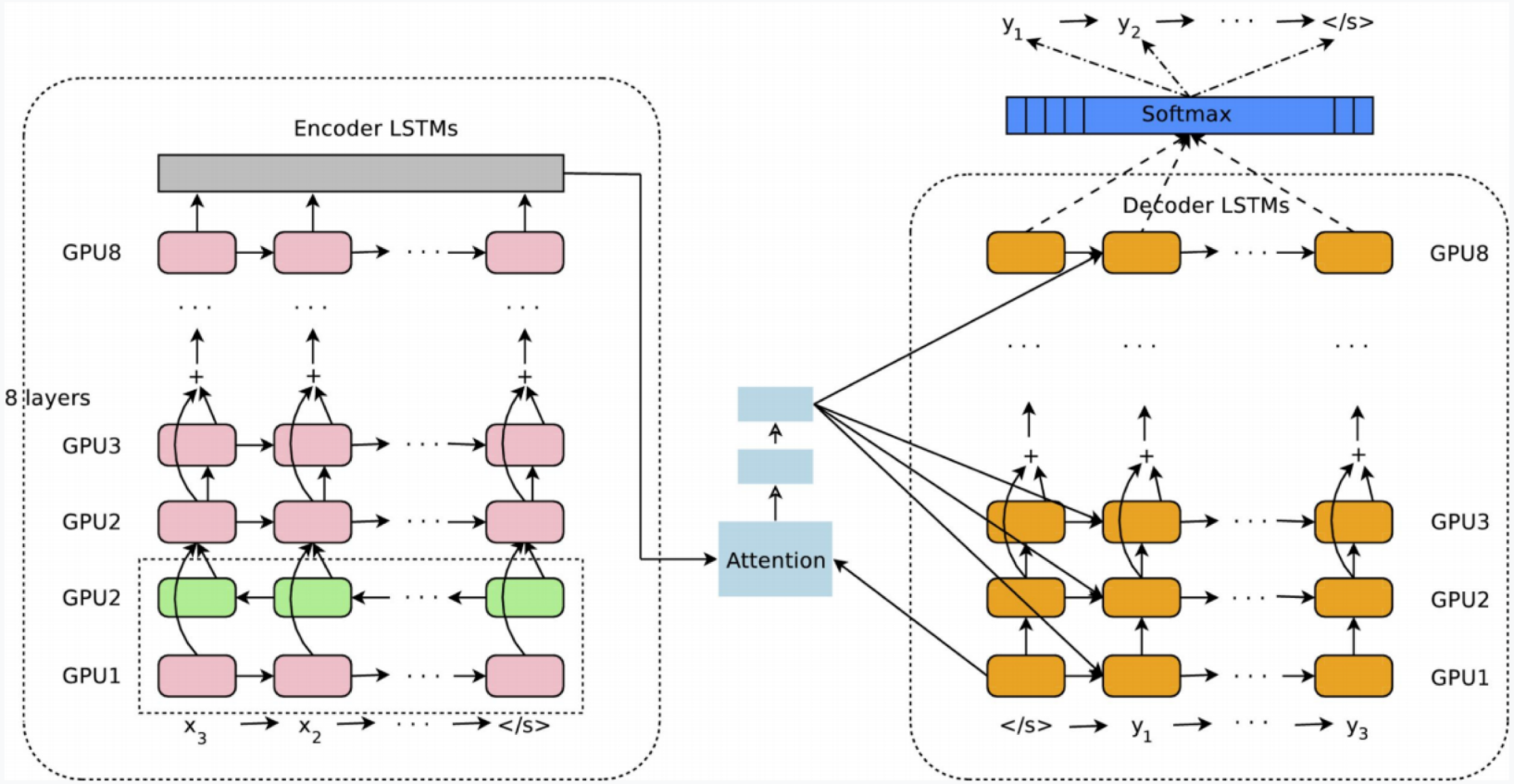


*Figure from "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation" Wu et al. 2016

ConvS2S - Gehring et al.

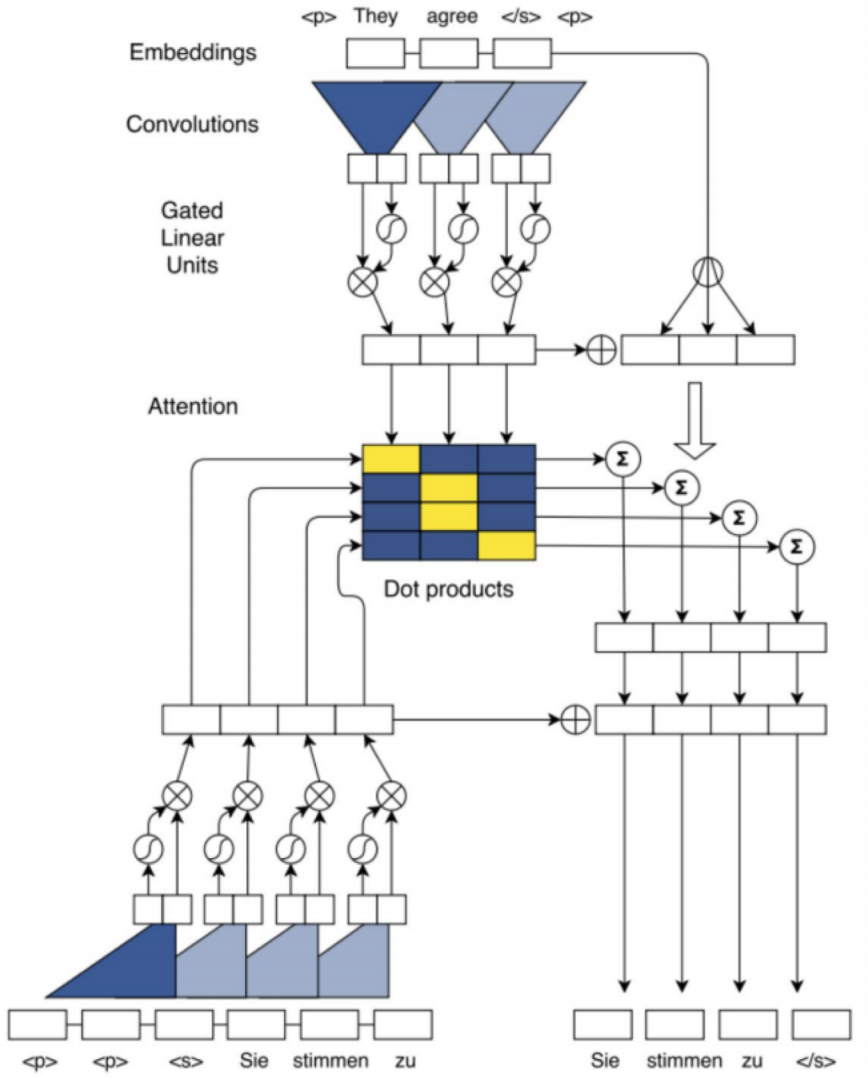


*Figure from "Convolutional Sequence to Sequence Learning" Gehring et al. 2017

- Core Components:
  - Convolution - GLUs
  - Multi-hop attention
  - Positional embeddings
  - Careful initialization
  - Careful normalization
  - Sync Training
- Pros:
  - No temporal dependence
  - More interpretable than RNN
- Cons:
  - Need to stack more to increase the receptive field

Transformer - Vaswani et al.

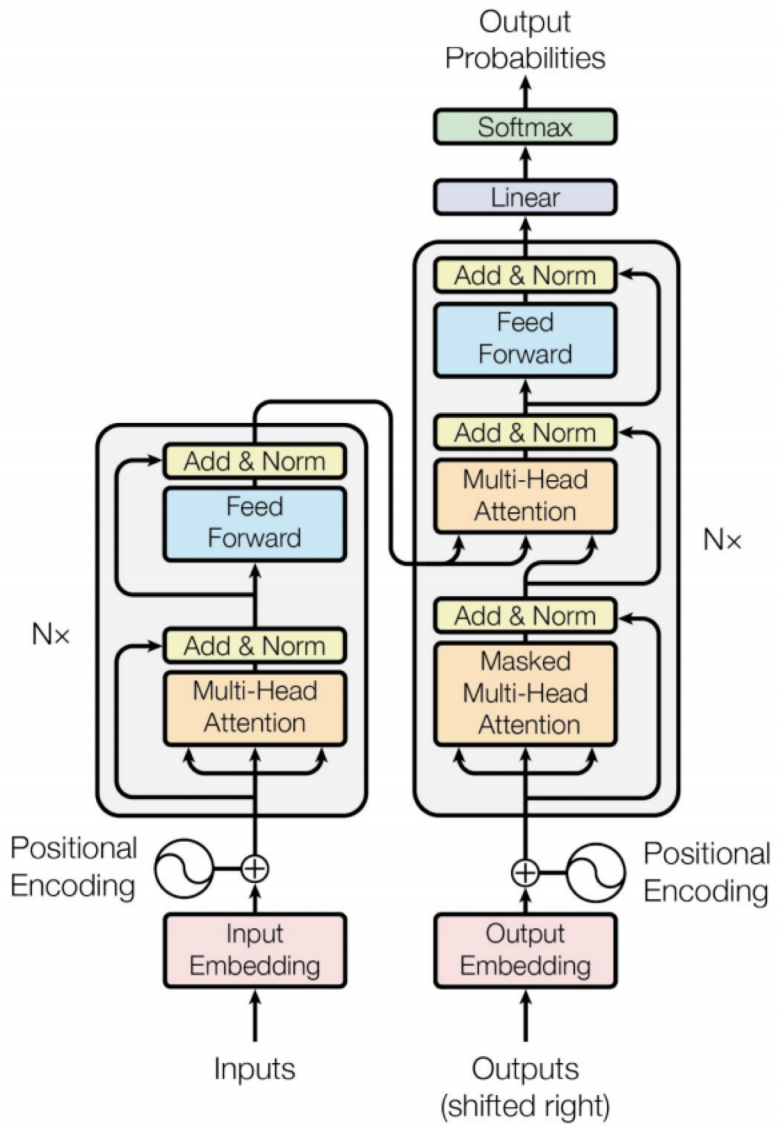
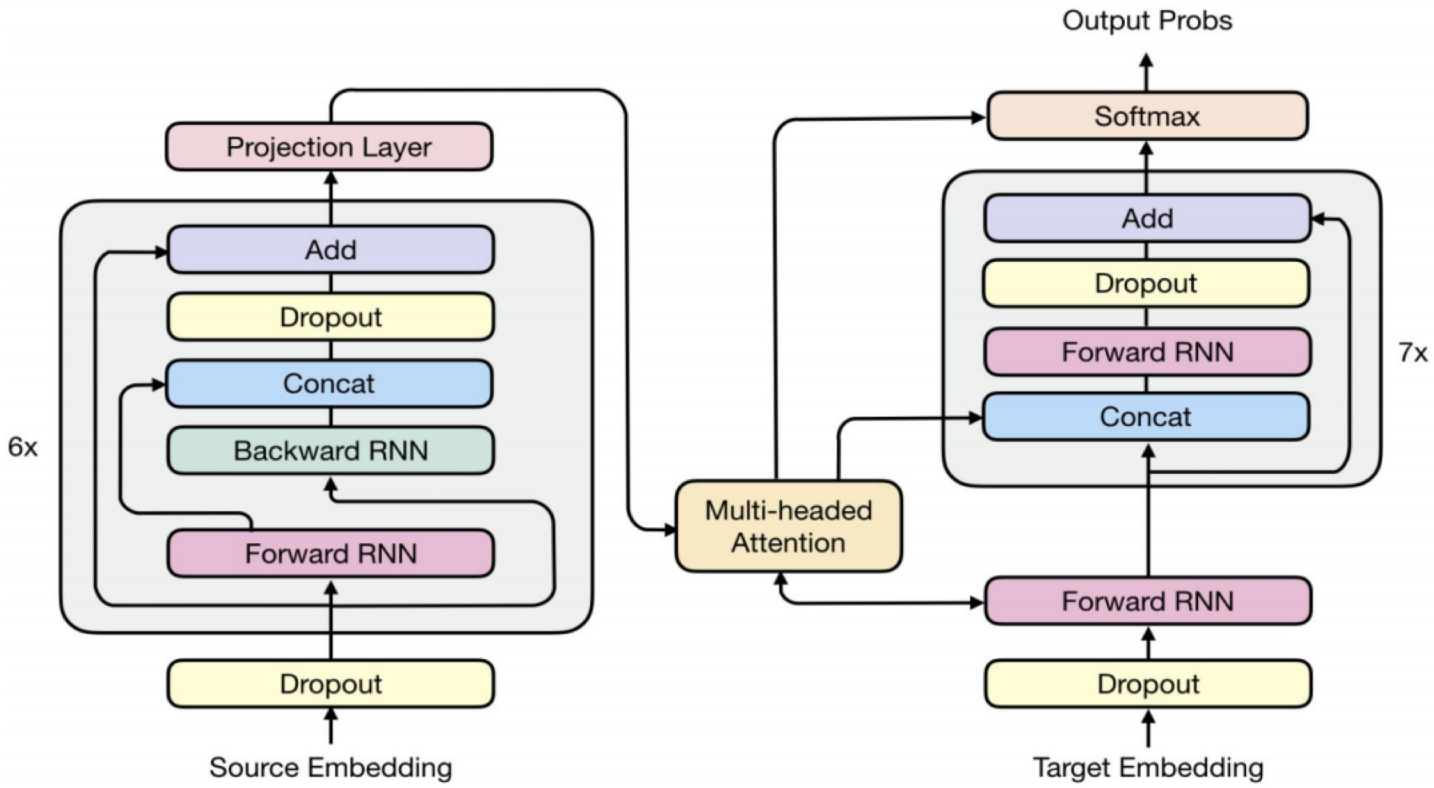


*Figure from "Attention is All You Need" Vaswani et al. 2017

Slide from Firat (2018) of Google AI

- Core Components:
  - Self-Attention
  - Multi-headed attention
  - Layout: N⟶ f() ⟶ D⟶ R
  - Careful normalization
  - Careful batching
  - Sync training
  - Label Smoothing
  - Per-token loss
  - Learning rate schedule
  - Checkpoint Averaging
- Pros:
  - Gradients everywhere - faster optimization
  - Parallel encoding both training/inference
- Cons:
  - Combines many advances at once
  - Fragile

The Best of Both Worlds - I: RNMT+



- The Architecture:
  - Bi-directional encoder 6 x LSTM
  - Uni-directional decoder 8 x LSTM
  - Layer normalized LSTM cell
    - Per-gate normalization
  - Multi-head attention
    - 4 heads
    - Additive (Bahdanau) attention

Model Comparison - I : BLEU Scores

WMT'14 En-Fr
(35M sentence pairs)

Model	Test BLEU	Epochs	Training Time
GNMT	38.95	-	-
ConvS2S [7]	39.49 ± 0.11	62.2	438h
Trans. Base	39.43 ± 0.17	20.7	90h
Trans. Big [8]	40.73 ± 0.19	8.3	120h
RNMT+	41.00 ± 0.05	8.5	120h

WMT'14 En-De
(4.5M sentence pairs)

Model	Test BLEU	Epochs	Training Time
GNMT	24.67	-	-
ConvS2S	25.01 ±0.17	38	20h
Trans. Base	27.26 ± 0.15	38	17h
Trans. Big	27.94 ± 0.18	26.9	48h
RNMT+	28.49 ± 0.05	24.6	40h

- RNMT+/ConvS2S: 32 GPUs, 4096 sentence pairs/batch.
- Transformer Base/Big: 16 GPUs, 65536 tokens/batch.

Model Comparison - II : Speed and Size

WMT'14 En-Fr
(35M sentence pairs)

Model	Test BLEU	Epochs	Training Time
GNMT	38.95	-	-
ConvS2S [7]	39.49 ± 0.11	62.2	438h
Trans. Base	39.43 ± 0.17	20.7	90h
Trans. Big [8]	40.73 ± 0.19	8.3	120h
RNMT+	41.00 ± 0.05	8.5	120h

Model	Examples/s	FLOPs	Params
ConvS2S	80	15.7B	263.4M
Trans. Base	160	6.2B	93.3M
Trans. Big	50	31.2B	375.4M
RNMT+	30	28.1B	378.9M

WMT'14 En-De
(4.5M sentence pairs)

Model	Test BLEU	Epochs	Training Time
GNMT	24.67	-	-
ConvS2S	25.01 ±0.17	38	20h
Trans. Base	27.26 ± 0.15	38	17h
Trans. Big	27.94 ± 0.18	26.9	48h
RNMT+	28.49 ± 0.05	24.6	40h

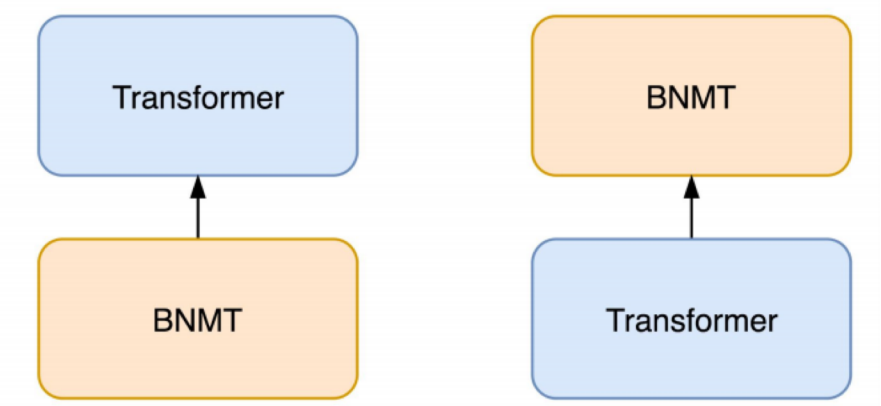
- RNMT+/ConvS2S: 32 GPUs, 4096 sentence pairs/batch.
- Transformer Base/Big: 16 GPUs, 65536 tokens/batch.

The Best of Both Worlds - II: Hybrids

Strengths of each architecture:

- **RNMT+**
  - Highly expressive - continuous state space representation.
- **Transformer**
  - Full receptive field - powerful feature extractor.
- Combining individual architecture strengths:
  - Capture complementary information - “Best of Both Worlds”.
- Trainability - important concern with hybrids
  - **Connections between different types of layers need to be carefully designed.**

Encoder - Decoder Hybrids



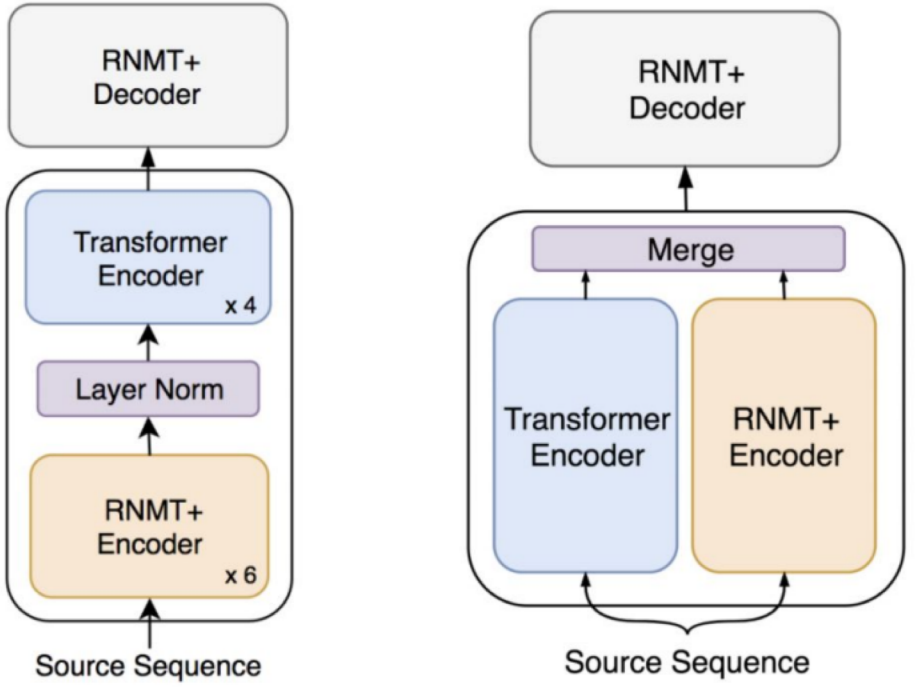
Encoder	Decoder	En→Fr Test BLEU
Trans. Big	Trans. Big	40.73 ± 0.19
RNMT+	RNMT+	41.00 ± 0.05
Trans. Big	RNMT+	$\mathbf{41.12 \pm 0.16}$
RNMT+	Trans. Big	39.92 ± 0.21

Separation of roles:

- Decoder - conditional LM
- Encoder - build feature representations

→ Designed to contrast the roles.
(last two rows)

Encoder Layer Hybrids



(a) Cascaded Encoder (b) Multi-Column Encoder

Improved feature extraction:

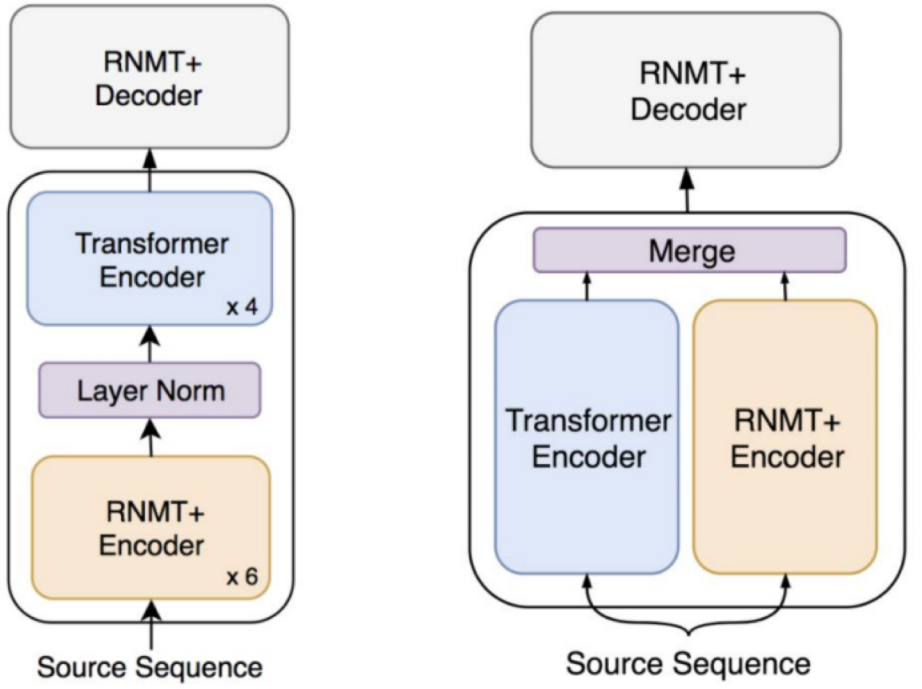
- Enrich stateful representations with global self-attention
- Increased capacity

Details:

- Pre-trained components to improve trainability
- Layer normalization at layer boundaries

Cascaded Hybrid - **vertical** combination
Multi-Column Hybrid - **horizontal** combination

Encoder Layer Hybrids



(a) Cascaded Encoder (b) Multi-Column Encoder

Model	En→Fr BLEU	En→De BLEU
Trans. Big	40.73 ± 0.19	27.94 ± 0.18
RNMT+	41.00 ± 0.05	28.59 ± 0.05
Cascaded	$\mathbf{41.67 \pm 0.11}$	28.62 ± 0.06
MultiCol	41.66 ± 0.11	$\mathbf{28.84 \pm 0.06}$

$$quality = f(X, \theta, \mu)$$

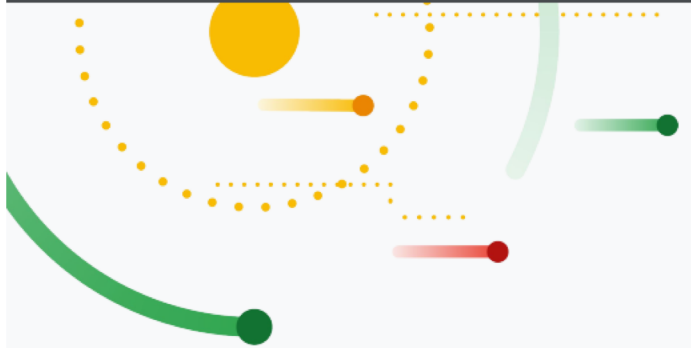
Modelling (expressivity)

Optimization (trainability)

X : Data

θ : Model

μ : Hyperparameters



$$quality = f(X, \theta, \mu)$$

Modelling (expressivity)

Optimization (trainability)

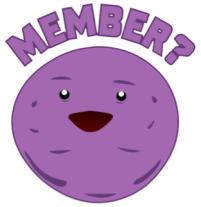
X : Data

θ : Model

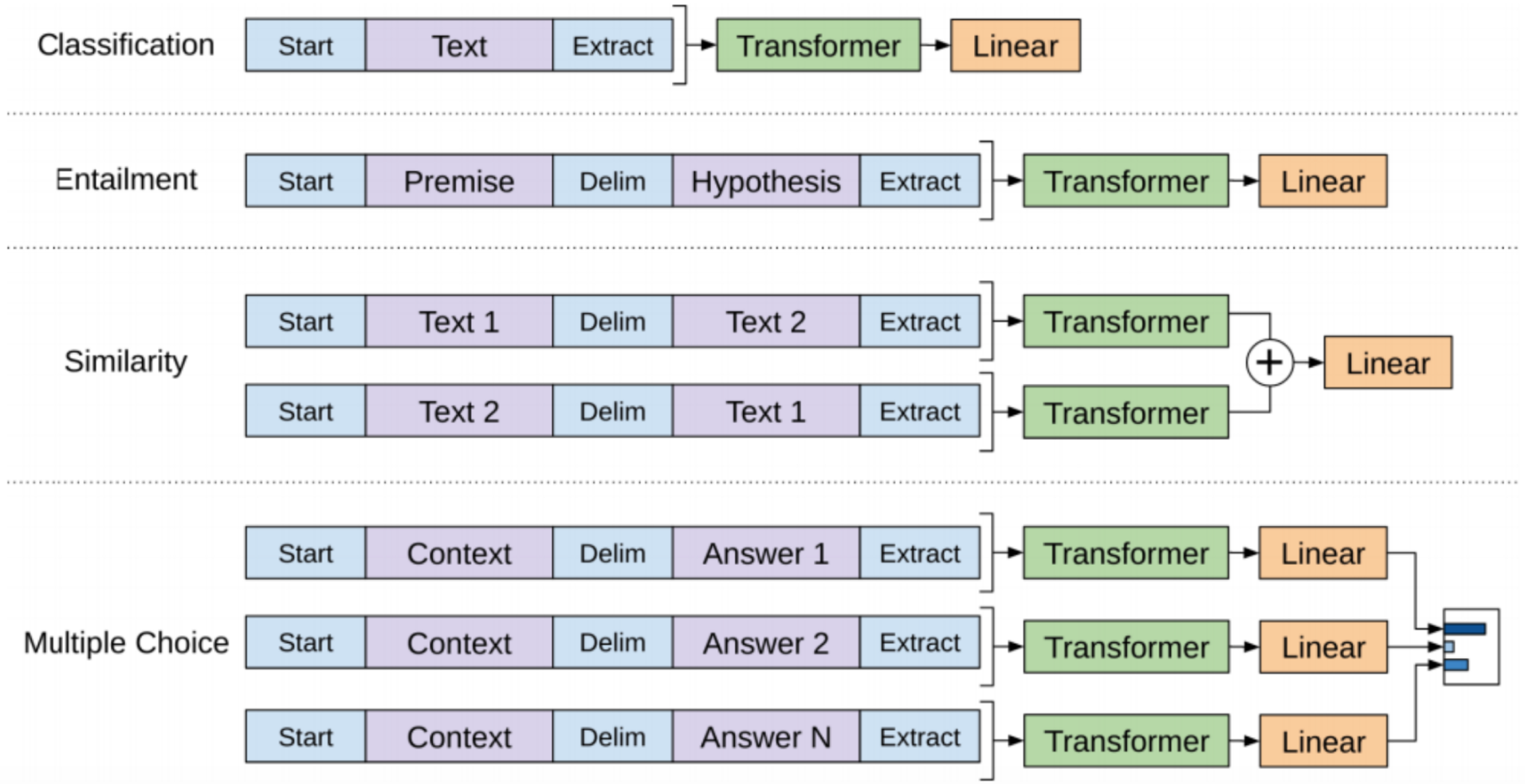
μ : Hyperparameters

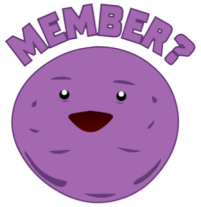
https://ufal.mff.cuni.cz/mtm18/files/04-tricks-of-the-trade-orhan-firat.pdf

Data Hacking



theses…





theses…

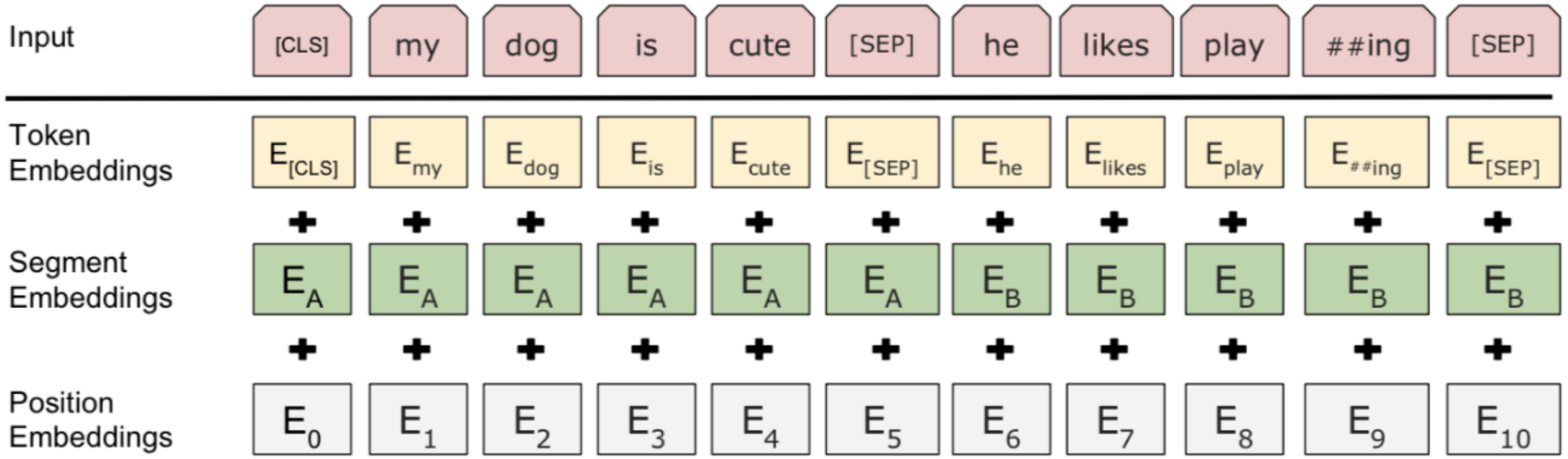
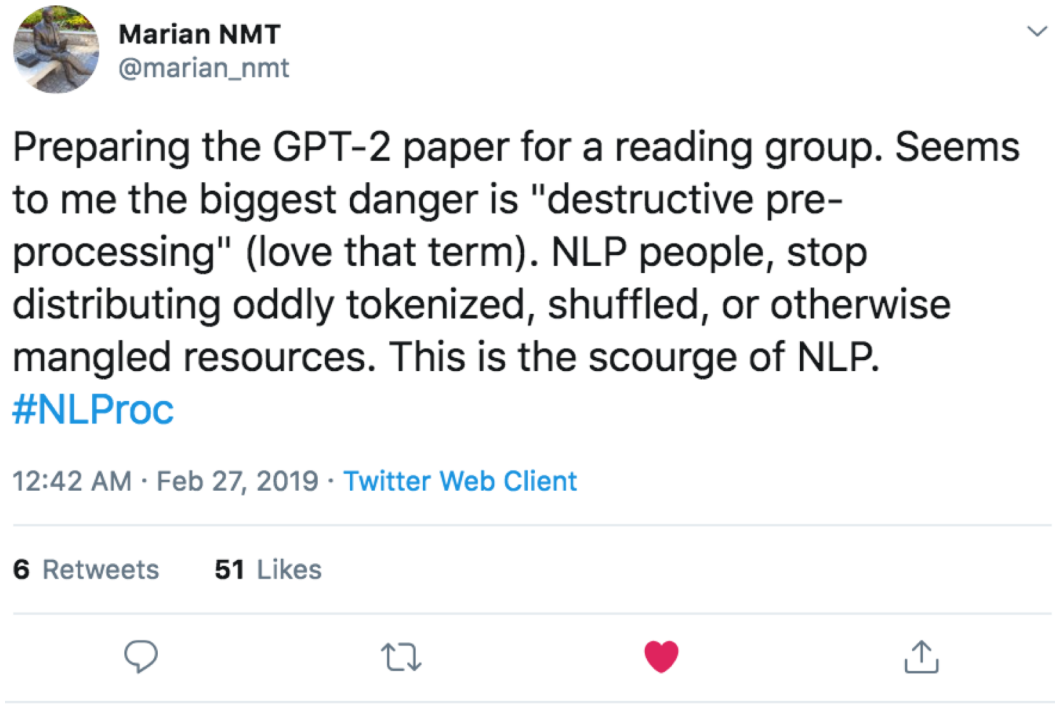


Fig. 11. BERT input representation. (Image source: original paper)

Note that the first token is always forced to be `[CLS]` — a placeholder that will be used later for prediction in downstream tasks.

Data Hacking



Byte Pair Encoding (Sennrich et al. 2016)

Algorithm 1 Learn BPE operations

```
import re, collections

def get_stats(vocab):
  pairs = collections.defaultdict(int)
  for word, freq in vocab.items():
    symbols = word.split()
    for i in range(len(symbols)-1):
      pairs[symbols[i],symbols[i+1]] += freq
  return pairs

def merge_vocab(pair, v_in):
  v_out = {}
  bigram = re.escape(' '.join(pair))
  p = re.compile(r'(?<!\S)' + bigram + r'(?!\S)')
  for word in v_in:
    w_out = p.sub(''.join(pair), word)
    v_out[w_out] = v_in[word]
  return v_out

vocab = {'l o w </w>' : 5, 'l o w e r </w>' : 2,
         'n e w e s t </w>':6, 'w i d e s t </w>':3}
num_merges = 10
for i in range(num_merges):
  pairs = get_stats(vocab)
  best = max(pairs, key=pairs.get)
  vocab = merge_vocab(best, vocab)
  print(best)
```

r · → r·
l o → lo
lo w → low
e r· → er·

WordPiece from GNMT (Wu et al. 2016)

4.1 Wordpiece Model

Our most successful approach falls into the second category (sub-word units), and we adopt the wordpiece model (WPM) implementation initially developed to solve a Japanese/Korean segmentation problem for the Google speech recognition system [35]. This approach is completely data-driven and guaranteed to generate a deterministic segmentation for any possible sequence of characters. It is similar to the method used in [38] to deal with rare words in Neural Machine Translation.

For processing arbitrary words, we first break words into wordpieces given a trained wordpiece model. Special word boundary symbols are added before training of the model such that the original word sequence can be recovered from the wordpiece sequence without ambiguity. At decoding time, the model first produces a wordpiece sequence, which is then converted into the corresponding word sequence.

Here is an example of a word sequence and the corresponding wordpiece sequence:

- **Word**: Jet makers feud over seat width with big orders at stake
- **wordpieces**: _J et _makers _fe ud _over _seat _width _with _big _orders _at _stake

SentencePiece (Kudo and Richardson, 2018)

```cpp
#include <sentencepiece_processor.h>
#include <sentencepiece_trainer.h>

SentencePieceTrainer::Train(
  "--input=input.txt "
  "--model_prefix=spm "
  "--vocab_size=1000");

SentencePieceProcessor sp;
sp.Load("spm.model");

std::vector<std::string> pieces;
sp.Encode("Hello world.", &pieces);

std::vector<int> ids;
sp.Encode("Hello world.", &ids);

std::string text;
sp.Decode({151, 88, 21, 887, 6}, &text);
```

Figure 3: C++ API usage (The same as Figure 1.)

```python
import sentencepiece as spm

params = ('--input=input.txt '
          '--model_prefix=spm '
          '--vocab_size=1000')
spm.SentencePieceTrainer.Train(params)

sp = spm.SentencePieceProcessor()
sp.Load('spm.model')

print(sp.EncodeAsPieces('Hello world.'))
print(sp.EncodeAsIds('Hello world.'))
print(sp.DecodeIds([151, 88, 21, 887, 6]))
```

Figure 4: Python API usage (The same as Figure 1.)

```python
import tensorflow as tf
import tf_sentencepiece as tfs

model = tf.gfile.GFile('spm.model','rb').read()

input_text = tf.placeholder(tf.string, [None])
ids, lens = tfs.encode(input_text, model_proto=model,
        out_type=tf.int32)
output_text = tfs.decode(ids, lens, model_proto=model)

with tf.Session() as sess:
   text = ['Hello world.','New York']
   ids_, lens_, output_text_ = sess.run([ids, lens, output_text
         ], feed_dict={input_text:text})
```

Figure 5: TensorFlow API usage

The SentencePiece model (model proto) is an attribute of the TensorFlow operation and embedded into the TensorFlow graph so the model and graph become purely self-contained.

```
>>> sp.Load('spm.model')
>>> for n in range(5):
... sp.SampleEncodeAsPieces('New York', -1, 0.1)
['_','N','e','w','_York']
['_','New','_York']
['_','New','_Y','o','r','k']
['_','New','_York']
['_','New','_York']
```

Figure 6: Subword sampling with Python API

SentencePiece (Kudo and Richardson, 2018)

Lang pair	setting (source/target)	# vocab.	BLEU
ja→en	Word model (baseline)	80k/80k	28.24
	SentencePiece	8k (shared)	29.55
	SentencePiece w/ pre-tok.	8k (shared)	29.85
	Word/SentencePiece	80k/8k	27.24
	SentencePiece/Word	8k/80k	29.14
en→ja	Word model (baseline)	80k/80k	20.06
	SentencePiece	8k (shared)	21.62
	SentencePiece w/ pre-tok.	8k (shared)	20.86
	Word/SentencePiece	80k/8k	21.41
	SentencePiece/Word	8k/80k	19.94

Table 1: Translation Results (BLEU(%))

Task	Tool	Pre-tok.	time (sec.) Japanese	time (sec.) English
Train	subword-nmt	yes	56.9	54.1
	SentencePiece	yes	10.1	16.8
	subword-nmt	no	528.0	94.7
	SentencePiece	no	217.3	21.8
Seg.	subword-nmt	yes	23.7	28.6
	SentencePiece	yes	8.2	20.3
	subword-nmt	no	216.2	36.1
	SentencePiece	no	5.9	20.3
Pre-tokenizaion KyTea(ja)/Moses(en)			24.6	15.8

Table 2: Segmentation performance. KFTT corpus (440k sentences) is used for evaluation. Experiments are executed on Linux with Xeon 3.5Ghz processors. The size of vocabulary is 16k. Moses and KyTea tokenizers are used for English and Japanese respectively. Note that we have to take the time of pre-tokenization into account to make a fair comparison with and without pre-tokenization. Because subword-nmt is based on BPE, we used the BPE model in SentencePiece. We found that BPE and unigram language models show almost comparable performance.

SentencePiece (Kudo and Richardson, 2018)

Comparisons with other implementations

Feature	SentencePiece	subword-nmt	WordPiece
Supported algorithm	BPE, unigram, char, word	BPE	BPE*
OSS?	Yes	Yes	Google internal
Subword regularization	Yes	No	No
Python Library (pip)	Yes	No	N/A
C++ Library	Yes	No	N/A
Pre-segmentation required?	No	Yes	Yes
Customizable normalization (e.g., NFKC)	Yes	No	N/A
Direct id generation	Yes	No	N/A

Note that BPE algorithm used in WordPiece is slightly different from the original BPE.

tRuEcasIng (Lita et al. 2003)

```
>>> from sacremoses import MosesTruecaser, MosesTokenizer

# Train a new truecaser from a 'big.txt' file.
>>> mtr = MosesTruecaser()
>>> mtok = MosesTokenizer()

# Save the truecase model to 'big.truecasemodel' using `save_to`
>> tokenized_docs = [mtok.tokenize(line) for line in open('big.txt')]
>>> mtr.train(tokenized_docs, save_to='big.truecasemodel')

# Save the truecase model to 'big.truecasemodel' after training
# (just in case you forgot to use `save_to`)
>>> mtr = MosesTruecaser()
>>> mtr.train('big.txt')
>>> mtr.save_model('big.truecasemodel')

# Truecase a string after training a model.
>>> mtr = MosesTruecaser()
>>> mtr.train('big.txt')
>>> mtr.truecase("THE ADVENTURES OF SHERLOCK HOLMES")
['the', 'adventures', 'of', 'Sherlock', 'Holmes']

# Loads a model and truecase a string using trained model.
>>> mtr = MosesTruecaser('big.truecasemodel')
>>> mtr.truecase("THE ADVENTURES OF SHERLOCK HOLMES")
['the', 'adventures', 'of', 'Sherlock', 'Holmes']
>>> print(mtr.truecase("THE ADVENTURES OF SHERLOCK HOLMES", return_str=True)
'the adventures of Sherlock Holmes'
```

This paper focuses on **truecasing**, which is the process of restoring case information to raw text. Besides text rEaDaBILiTY, truecasing enhances the quality of case-carrying data, brings into the picture new corpora originally considered too noisy for various NLP tasks, and performs case normalization across styles, sources, and genres.

tRuEcasIng (Lita et al. 2003)

		BLEU Breakdown			
System	BLEU	1gr Precision	2gr Precision	3gr Precision	4gr Precision
all lowercase	0.1306	0.6016	0.2294	0.1040	0.0528
rule based	0.1466	0.6176	0.2479	0.1169	0.0627
1gr truecasing	0.2206	0.6948	0.3328	0.1722	0.0988
1gr truecasing+	0.2261	0.6963	0.3372	0.1734	0.0997
lm truecasing	0.2596	0.7102	0.3635	0.2066	0.1303
lm truecasing+	**0.2642**	**0.7107**	**0.3667**	**0.2066**	**0.1302**

Table 1: BLEU score for several truecasing strategies. (*truecasing*+ methods additionally employ the "first sentence letter uppercased" rule adjustment).

Little Things Goes a Long Way

moses-smt / mosesdecoder

Unwatch 161 | Unstar 923 | Fork 549

Code | Pull requests 0 | Projects 0 | Insights

Fix non-ASCII lowercasing #209

Edit

Merged **hieuhoang** merged 1 commit into `moses-smt:master` from `joelb-git:multi-bleu-detok-non-ascii-fix` 11 days ago

Conversation 1 | Commits 1 | Checks 0 | Files changed 1

Changes from **all commits** | File filter... | Jump to... | +3 −0

Diff settings | Review changes

3 scripts/generic/multi-bleu-detok.perl — Copy path | View file

```
@@ -14,6 +14,9 @@
 use warnings;                          14  use warnings;
 use strict;                            15  use strict;
                                        16
                                        17 +binmode(STDIN, ":utf8");
                                        18 +use open ':encoding(UTF-8)';
                                        19 +
 my $lowercase = 0;                     20  my $lowercase = 0;
 if ($ARGV[0] eq "-lc") {               21  if ($ARGV[0] eq "-lc") {
   $lowercase = 1;                      22    $lowercase = 1;
```

Little Things Goes a Long Way

joelb-git commented 14 days ago — Contributor

`multi-bleu-detok.perl` -lc does not lowercase non-ASCII characters.

Here's a tiny test case:

```
$ cat ref
ты была непримирима.
но что беспокоит меня, так это то, что...

$ cat hyp
Ты был недоказательным.
Но для меня это то, что
```

Before the change:

```
$ cat hyp | scripts/generic/multi-bleu-detok.perl -lc ref
BLEU = 18.50, 54.5/33.3/28.6/20.0 (BP=0.580, ratio=0.647, hyp_len=11, ref_len=17)
```

After:

```
$ cat hyp | scripts/generic/multi-bleu-detok.perl -lc ref
BLEU = 19.88, 72.7/33.3/28.6/20.0 (BP=0.580, ratio=0.647, hyp_len=11, ref_len=17)
```

 Fix non-ASCII lowercasing — fdb7384

  **hieuhoang** merged commit **187a75c** into `moses-smt:master` 11 days ago — Revert

Machine Translation: The Hunger (for BLEU) Games

ACL 2019
FOURTH CONFERENCE ON
MACHINE TRANSLATION (WMT19)

August 1-2, 2019
Florence, Italy

Home

[HOME]
TRANSLATION TASKS: [NEWS] [BIOMEDICAL] [ROBUSTNESS] [SIMILAR]
EVALUATION TASKS: [METRICS] [QUALITY ESTIMATION]
OTHER TASKS: [AUTOMATIC POST-EDITING] [PARALLEL CORPUS FILTERING]

This conference builds on a series of annual workshops and conferences on statistical machine translation, going back to 2006:

- the NAACL-2006 Workshop on Statistical Machine Translation,
- the ACL-2007 Workshop on Statistical Machine Translation,
- the ACL-2008 Workshop on Statistical Machine Translation,
- the EACL-2009 Workshop on Statistical Machine Translation,
- the ACL-2010 Workshop on Statistical Machine Translation
- the EMNLP-2011 Workshop on Statistical Machine Translation,
- the NAACL-2012 Workshop on Statistical Machine Translation,
- the ACL-2013 Workshop on Statistical Machine Translation,
- the ACL-2014 Workshop on Statistical Machine Translation,
- the EMNLP-2015 Workshop on Statistical Machine Translation,
- the First Conference on Machine Translation (at ACL-2016),
- the Second Conference on Machine Translation (at EMNLP-2017),
- the Third Conference on Machine Translation (at EMNLP-2018).

WMT 2018

Third Conference on Machine Translation

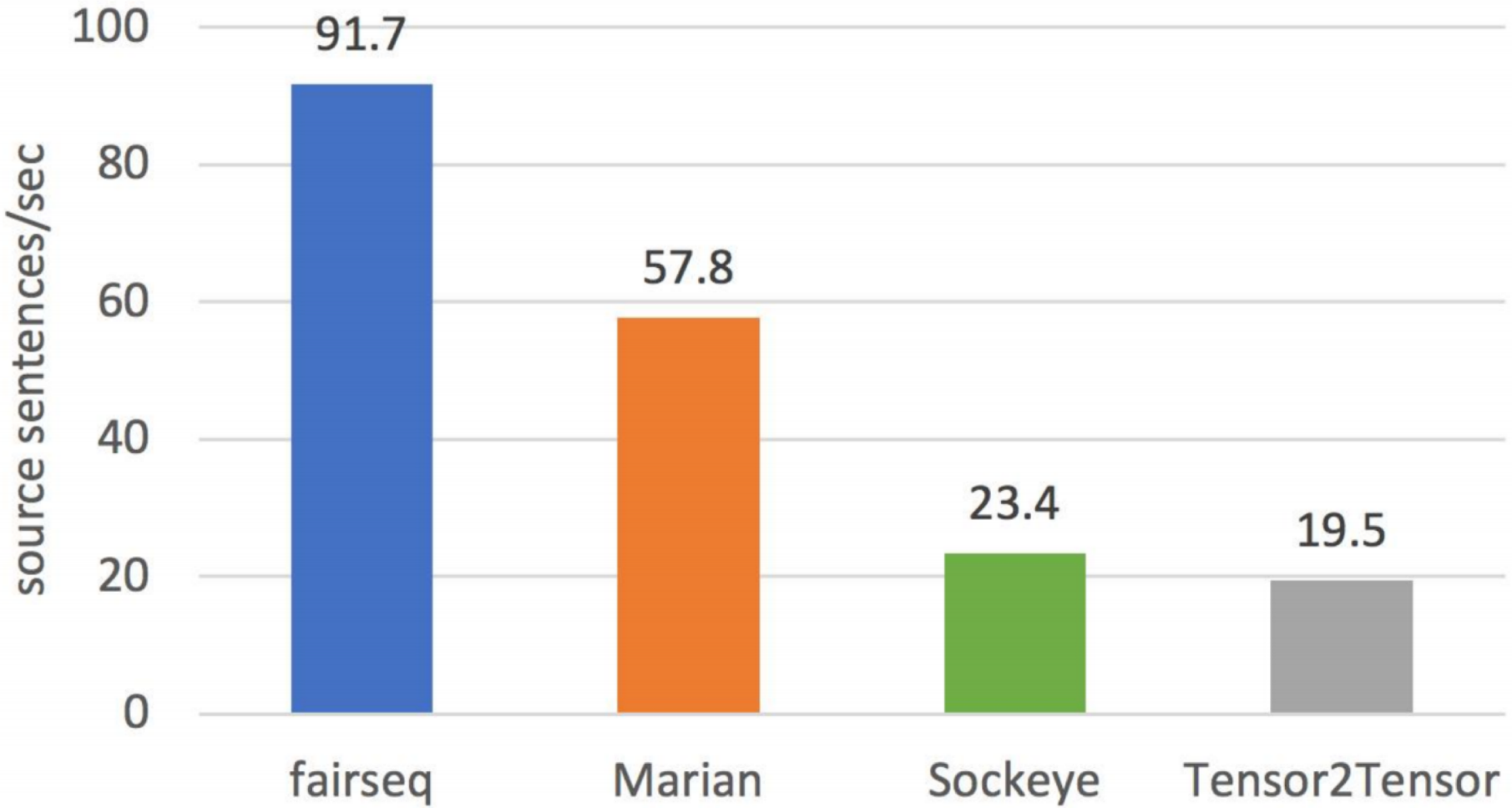
Evaluation Matrix

Number of systems for test set newstest2018

Translations | Resources | Download | Info | Account

input language \ output language	Czech	German	English	Estonian	Finnish	Russian	Turkish	Chinese
Czech			3 systems					
German			17 systems					
English	6 systems	25 systems		21 systems	19 systems	9 systems	14 systems	33 systems
Estonian			25 systems					
Finnish			17 systems					
Russian			7 systems					
Turkish			10 systems					
Chinese			20 systems					

Yes, it can get very competitive…


100
80
60
40
20
0
source sentences/sec
91.7
57.8
23.4
19.5
fairseq
Marian
Sockeye
Tensor2Tensor


Michael Auli
June 15 ·
We are releasing new features for fairseq, FAIR's sequence to sequence learning library:
https://github.com/pytorch/fairseq
Distributed training, fp16, delayed batching
We release code and pre-trained models to reproduce our recent paper "Scaling Neural Machine Translation" (https://arxiv.org/abs/1806.00187) where we train on up to 128 GPUs with half precision floating point operations as well ... See More
133
49 Shares
Like
Comment
Share
Write a comment...


James Bradbury
@jekbradbury
Following
Facebook's fairseq MT engine is really, really fast... Like, 50% faster than @marian_nmt (which is itself way faster than Sockeye/OpenNMT/Tensor2Tensor/xnmt/Nematus/etc) at generating from the same Transformer model
facebook.com/61013326/posts ...
2:24 PM - 15 Jun 2018
37 Retweets 139 Likes
2
37
139
Tweet your reply
Marian NMT @marian_nmt · Jun 15
Replying to @jekbradbury
Hold my beer ;)
1
13

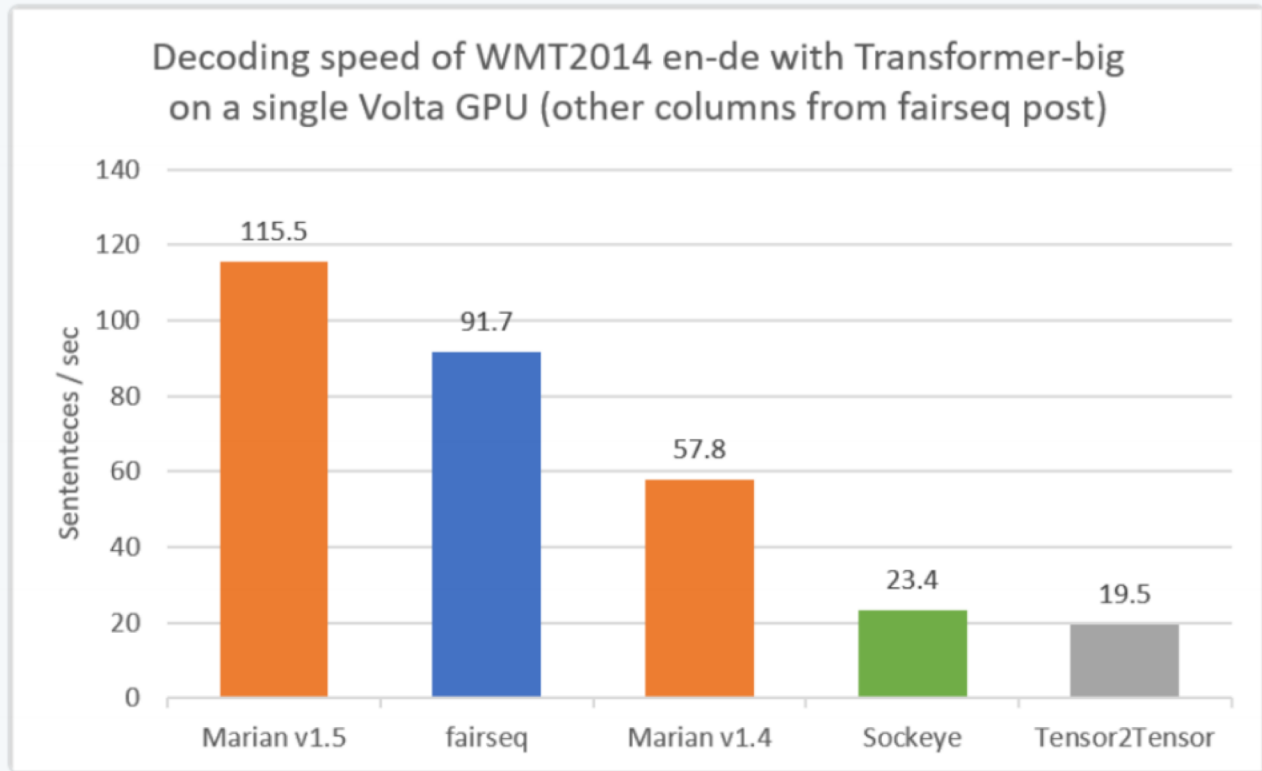
Marian NMT @marian_nmt · Jun 17

Boom! Marian v1.5.0 released.

Includes:

- Extensions from the WNMT shared task on efficiency arxiv.org/abs/1805.12096
- Optimized GPU-decoding for Transformer models.

See chart below for speed comparison to v1.4.0 (based on FAIR's post) @jekbradbury @alvations



4 33 108

We'll look at some of these tools in the hands-on today.

NUS National University of Singapore | iss INSTITUTE OF SYSTEMS SCIENCE

Wait a minute…

Isn't Google Translate free?

Why would companies care about Machine Translation?

Google Translate Pricing

Prices per month

	0-1 billion characters
Translation	$20 per 1,000,000 characters*
Language Detection	$20 per 1,000,000 characters*

If you pay in a currency other than USD, the prices listed in your currency on Cloud Platform SKUs apply.

* Price is per character sent to the API for processing, including whitespace characters. Empty queries are charged for one character. Google charges on per character basis, even if the character is multiple bytes, where a character corresponds to a (code-point). For example, translating "こんにちは" to English counts as 5 characters for the purposes of billing.

Amazon Translate Pricing

Pricing

$15 PER MILLION CHARACTERS

You are billed monthly for the total number of characters sent to the API for processing, including whitespace characters. Amazon Translate is priced at $15 per million characters ($0.000015 per character).

Free Tier

2 MILLION CHARACTERS PER MONTH FOR 12 MONTHS

The Free Tier is available to you for 12 months, starting from the date on which you create your first translation request. When your free usage expires, or if your application use exceeds the free usage tier, you simply pay standard, pay-as-you-go service rates.

Microsoft Translate Pricing

Select Offer: Translator Text

Region: Central US

Currency: US Dollar ($)

Pay-as-you-go

S1	Standard Translation Text Translation	$10 per million chars of standard translation
	Language Detection	
	Bilingual Dictionary	
	Transliteration	
	Custom Translation Translation	$40 per million chars of custom translation
	Training	$10 per million source + target chars of training data (max. $300/training)
	Custom model hosting	$10 per hosted custom translation model per region, per month

Commercial Demand

"Facebook is now serving 2 billion text translations per day. ... when we turned it off for some people, they went nuts!" - Josh Constine (2016)

"Machine translation at eBay is key in promoting cross-border trade ... Our buyers can search the site in their native language, but see inventory from far away. " - Evgeny Matusov (2016)

"To help our government customers and organizations in government-regulated industries, we are making Amazon Translate available in the AWS GovCloud (US) Region, Amazon's isolated cloud region built for sensitive data and regulated workloads" - Woo Kim (2018)

Commercial Demand

Leveraging on Technology

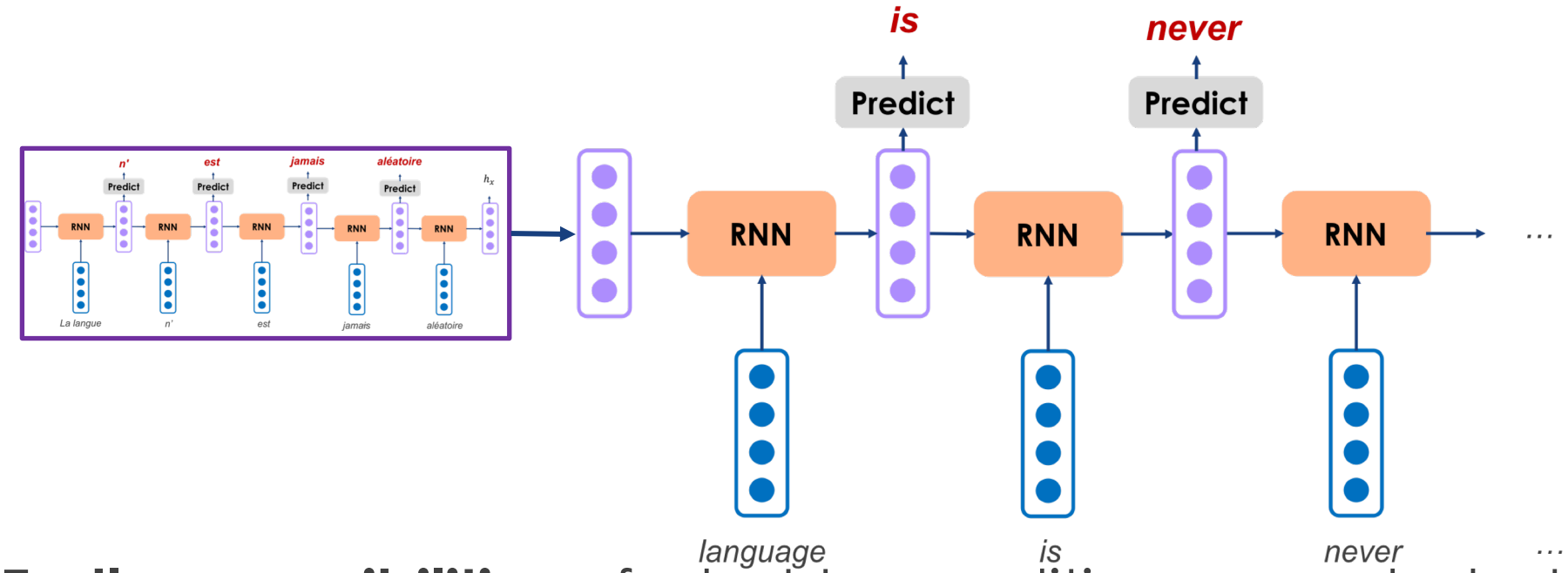
13. Next, I will speak on the 2nd "T", which is Technology. As you may be aware, the last two years have seen many game-changing breakthroughs in translation technology. Many big tech companies are investing heavily in this area: products like Google Pixel buds and many speech-to-speech translation gadgets and apps developed by Chinese and Japanese companies available to break down language barriers in communications. There are also translation tools equipped with Neural Technology, Artificial Intelligence and machine learning capabilities which could produce more accurate translations.

14. Last year, I informed everyone that MCI has embarked on a collaboration with A*STAR to develop a Customised Government Machine Translation Engine. The project is making good progress in performing English-Chinese pair translation. The researchers did a demonstration at the last NTC meeting. Preliminary assessments show that this engine is able to yield a higher BLEU[1] score compared to commercially available translation engines. The higher the BLEU score, the closer the machine output is compared to a professional human translator. The team will continue to train the engine for higher efficiency and accuracy, and we hope that by the middle of this year, a beta version can be deployed for the public to test out English-Chinese translations of common government phrases. The researchers have also started work to develop similar engines for English-Malay and English-Tamil language pairs. We look forward to the successful launch of these engines.

From Singapore's Ministry of Communication and Information (Jan 2018)

Unsupervised NMT

Sequence-to-Sequence

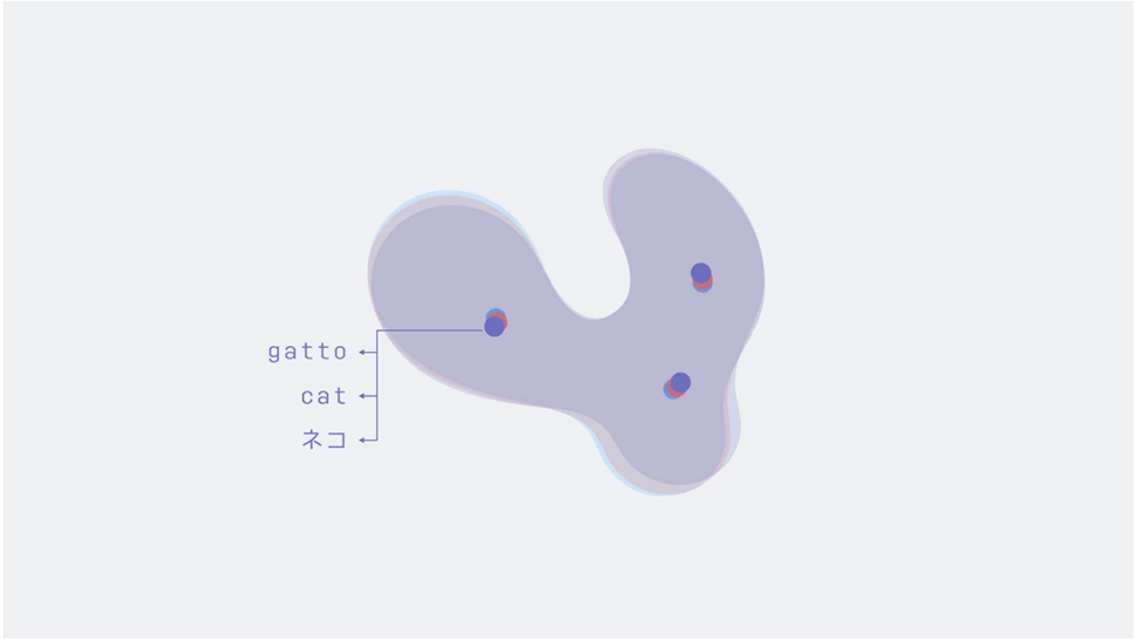


- **Endless possibilities** of what to condition on and what to generate
- But training requires the **paired condition and target generation**
- And relatively **large amount of data is needed** for model to train well

Machine Translation Achieved Human Parity

Algorithm 1: Unsupervised MT

1 **Language models:** Learn language models P_s and P_t over source and target languages;
2 **Initial translation models:** Leveraging P_s and P_t, learn two initial translation models, one in each direction: $P^{(0)}_{s\to t}$ and $P^{(0)}_{t\to s}$;
3 **for** k=1 **to** N **do**
4 **Back-translation:** Generate source and target sentences using the current translation models, $P^{(k-1)}_{t\to s}$ and $P^{(k-1)}_{s\to t}$, factoring in language models, P_s and P_t;
5 Train new translation models $P^{(k)}_{s\to t}$ and $P^{(k)}_{t\to s}$ using the generated sentences and leveraging P_s and P_t;
6 **end**



(*Image from Facebook Code Blog)

"Two-dimensional word embeddings in two languages can be aligned via a simple rotation. After the rotation, word translation is performed via nearest neighbor search."
– *Facebook Code Blog*

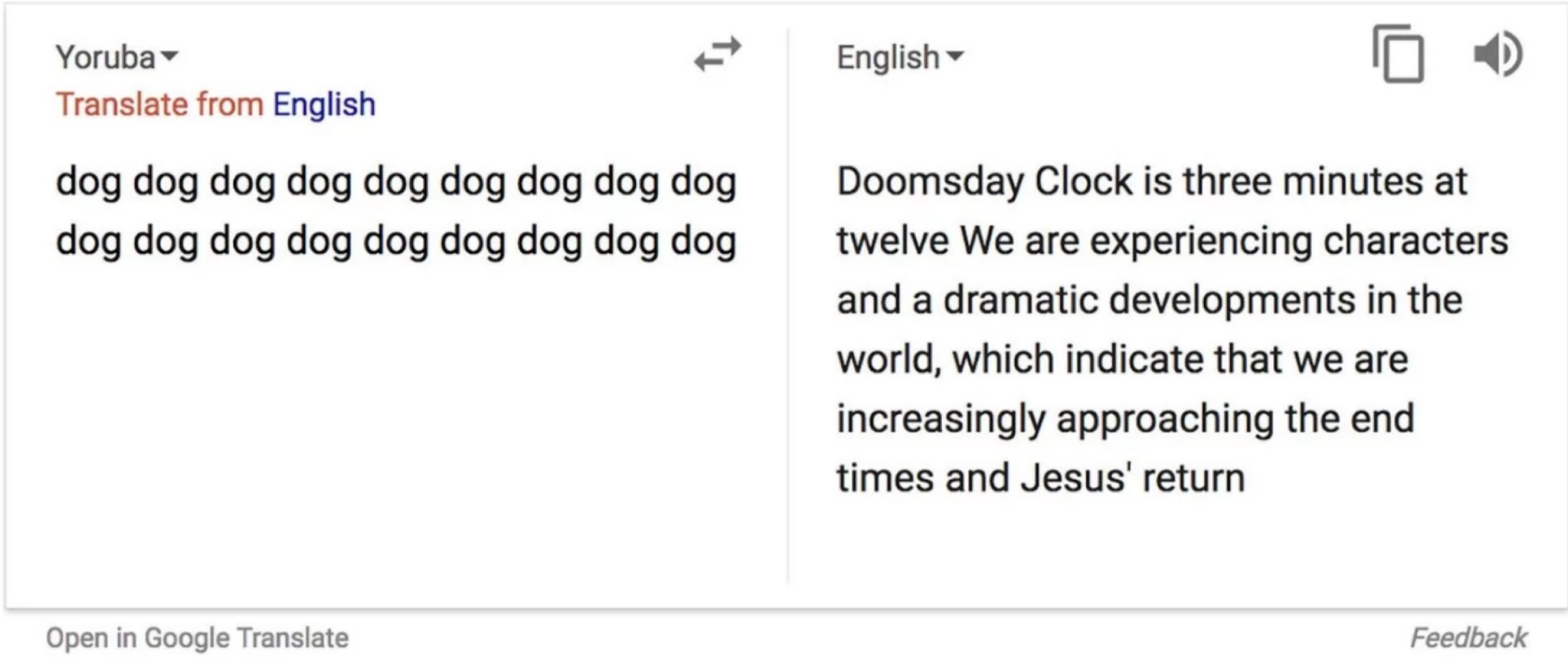
Unsupervised Machine Translation

- **Unsupervised SMT (IXA NLP Group)**
  - Non-neural: http://aclweb.org/anthology/D18-1399
- **Unsupervised NMT (IXA NLP and Cho)**
  - https://arxiv.org/pdf/1710.11041.pdf
- **Unsupervised NMT with weights sharing (UCAS)**
  - http://www.aclweb.org/anthology/P18-1005
- **Lots of unsupervised MT tech at Facebook at EMNLP 2018**
  - https://research.fb.com/facebook-research-at-emnlp/

Neural MT Fail #emptyneuron

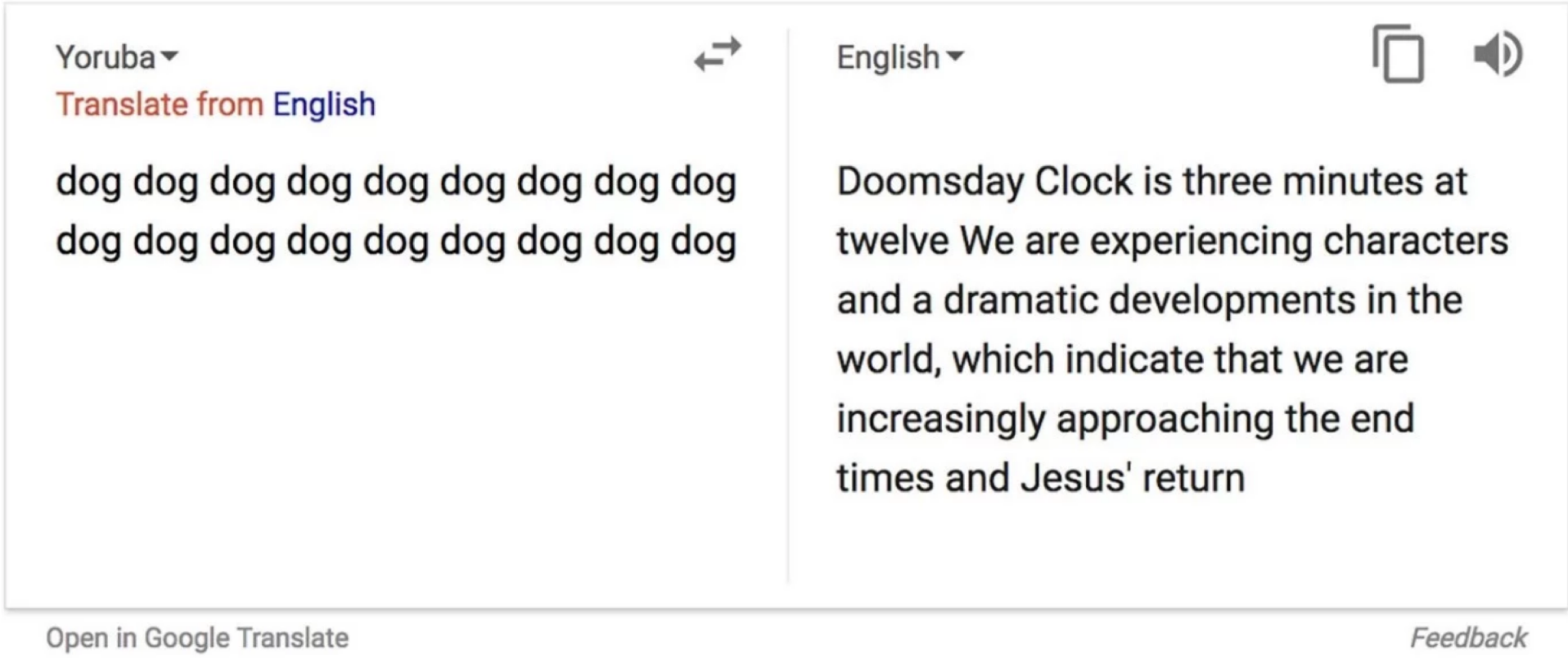
Neural MT Fail #emptyneuron

If you entered the word "dog" 20 times repeatedly—and translated it from a language like Yoruba—the tool returned this:



Neural MT Fail #emptyneuron

If you entered the word "dog" 20 times repeatedly—and translated it from a language like Yoruba—the tool returned this:



Neural MT Fail #emptyneuron

Netanyahu meant to say "Netta, you're a real darling," using the word *kapara*. In Hebrew, the term is often used as slang for affection or blessing. But it also contains the three Hebrew letters that spell "cow," which, translated without context, could be rendered into "like a cow."

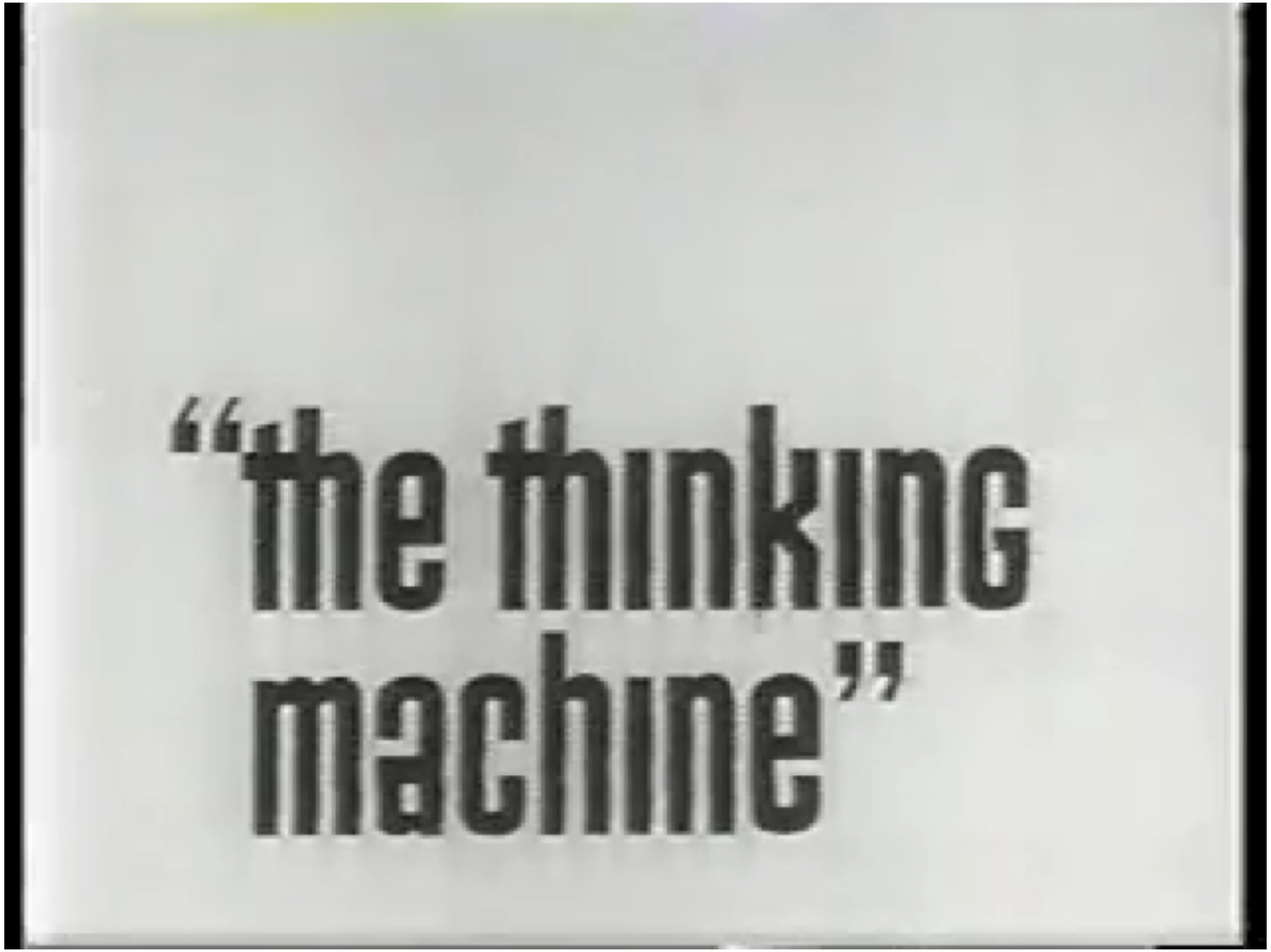
Neural MT Fail #emptyneuron

During the 2018 Winter Olympics in PyeongChang, they used Google Translate to place an order with a local supermarket for 1,500 eggs. Somehow, an extra "0" got added, which might have had something to do with how similar 1,500 and 15,000 look in Korean:

1,500 일천오백
15,000 일만오천

Managing Expectations

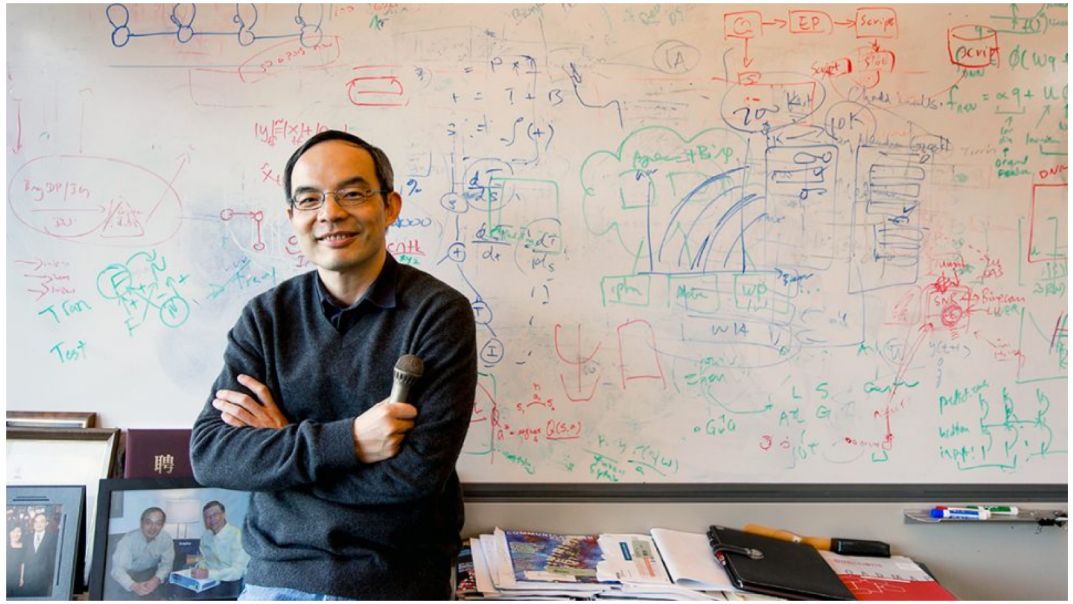
Managing Expectation



"We should be able to do, with a modern computer, about 1 to 2 million words an hour. This would be quite an adequate speed to cope with the whole output of the Soviet Union. It's just a few hours computer time a week ... ***If our experiment goes well,*** *[we'll be able to achieve the speed],* ***in about 5 years or so.***

Yes, [it means the end of human translators], for translators for scientific and technical material but as regards poetry and novel, no I don't think we'll ever replace these translators"

Machine Translation Achieved Human Parity



(*Image from Microsoft AI Blog)

"Microsoft reaches a historic milestone, using AI to match human performance in translating news from Chinese to English"
- Microsoft AI Blog

Definition of Human Parity

If there is ***no statistically significant difference*** between human quality scores for … machine translation … and the scores for the corresponding human translations then the machine has achieved human parity.

Did MT really achieve Human Parity?

- **Has Machine Translation Achieved Human Parity? A Case for Document-level Evaluation**
  - Document level evaluation is necessary
  - http://aclweb.org/anthology/D18-1512

- **What Level of Quality can Neural Machine Translation Attain on Literary Text?**
  - Only 17-34% of novels can be machine translated
  - https://link.springer.com/chapter/10.1007/978-3-319-91241-7_12

Did MT really achieve Human Parity?

- **Quality expectations of machine translation**
  - *"those of us who have seen many paradigms come and go know that overgilding the lily does none of us any good"*
  - https://arxiv.org/pdf/1803.08409.pdf

- **"Human parity" in machine translation**
  - *"Some aspects of such translations are very good, but the frequent mistakes spoil things."*
  - http://languagelog.ldc.upenn.edu/nll/?p=40602

Hands-on:
Now, lets see some action!

Remember this?

That's fun but it's just a toy for me to apply what I learn in theory and get my hands dirty.

Also, a simple modelling (without optimization) is good for educational purposes.

And now…

- **AllenNLP:** http://www.realworldnlpbook.com/blog/building-seq2seq-machine-translation-models-using-allennlp.html
- **Fairseq:** https://github.com/pytorch/fairseq/blob/master/examples/translation/README.md
- **Marian:** https://marian-nmt.github.io/examples/mtm2018-labs
- **OpenNMT:** http://opennmt.net/OpenNMT-py/quickstart.html#
- **Many more on** https://github.com/jonsafari/nmt-list